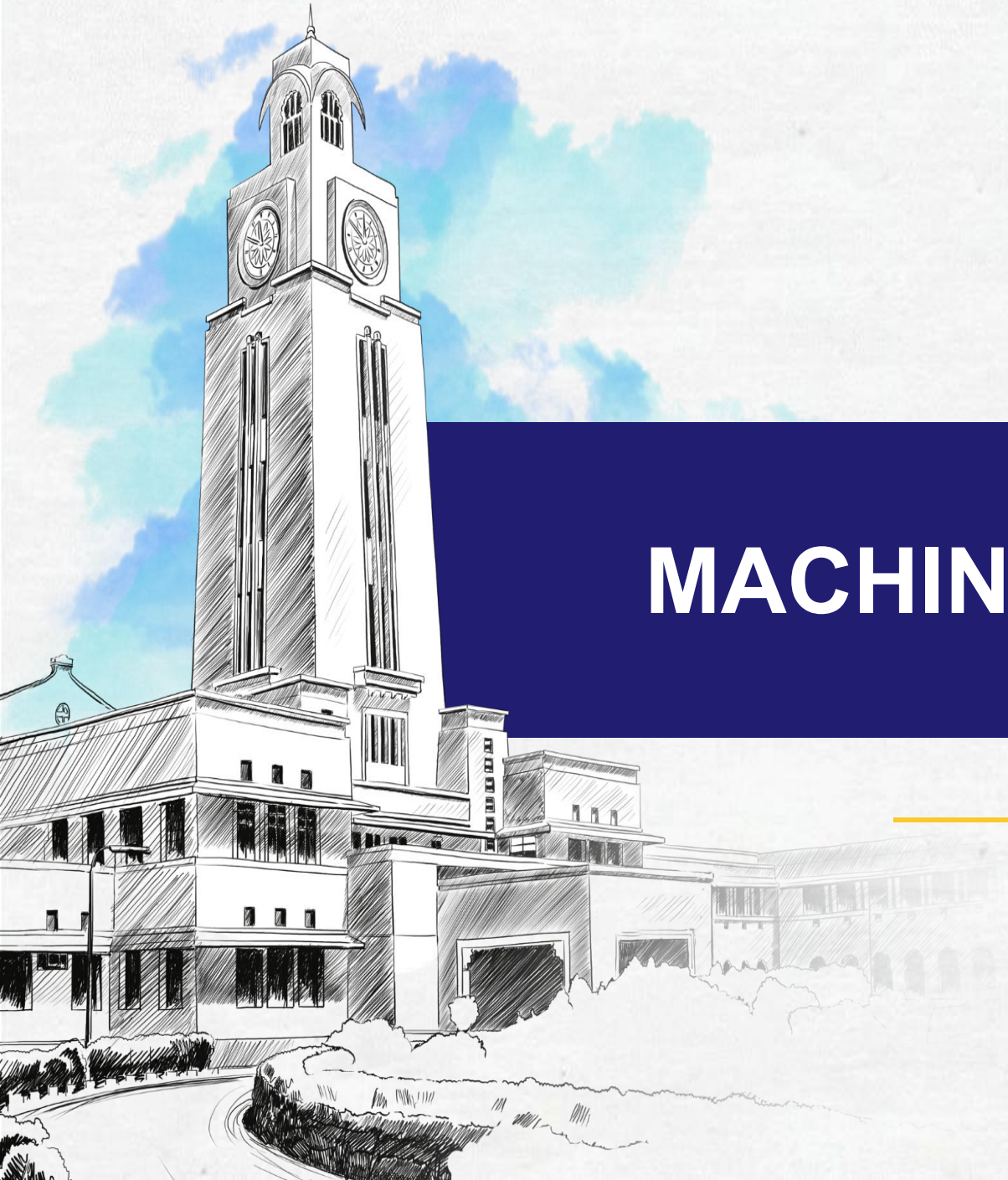
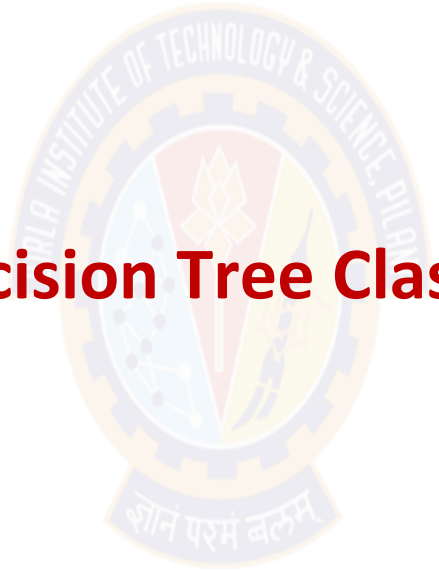


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

MACHINE LEARNING

Session 7 & 8
(8th March 2026)

Agenda



Decision Tree Classifier

Course Plan

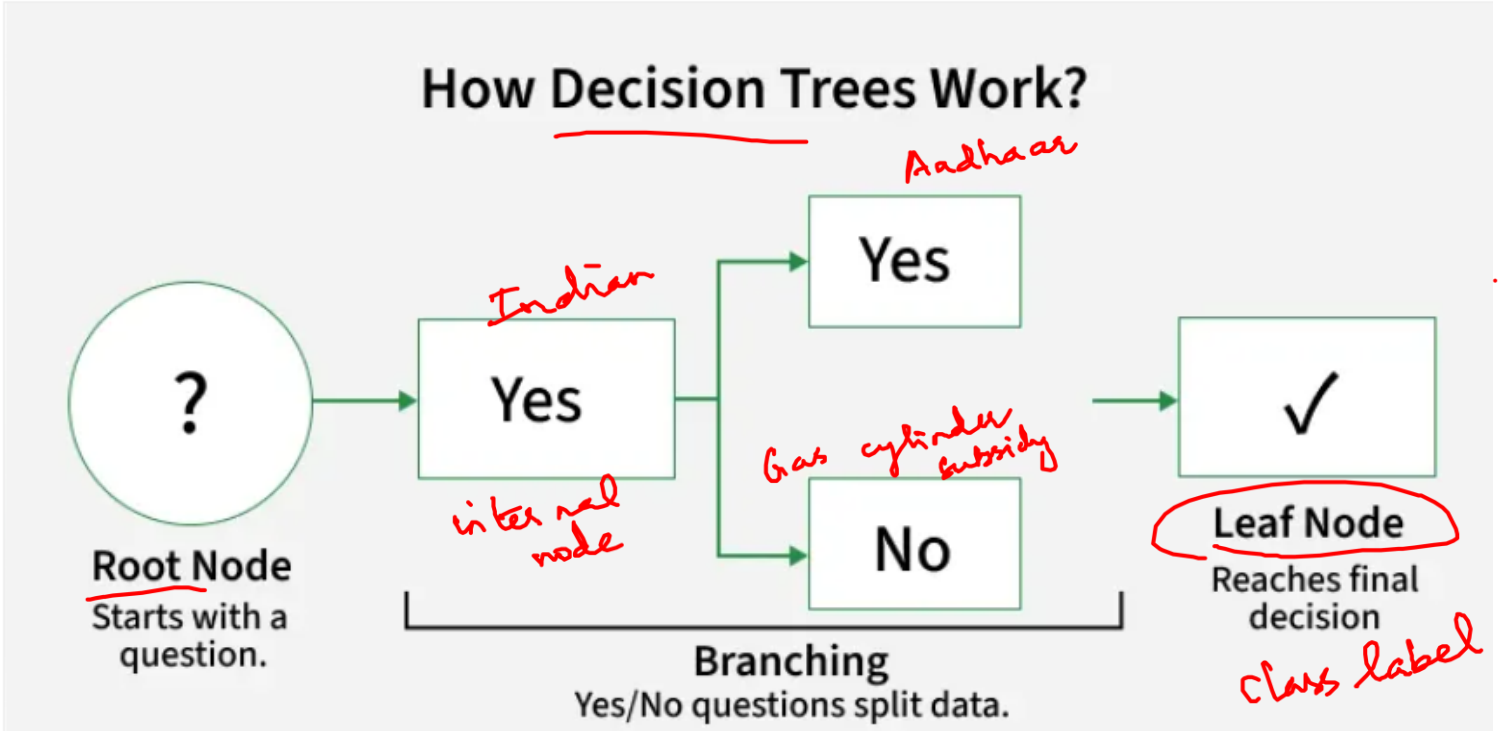
M1	Introduction
M2	Machine learning Workflow
M3	Linear Models for Regression
M4	Linear Models for Classification
M5	Decision Tree
M6	Instance Based Learning
M7	Support Vector Machine
M8	Bayesian Learning
M9	Ensemble Learning
M10	Unsupervised Learning
M11	Machine Learning Model Evaluation/Comparison

Module 5

- Information Theory
- Entropy Based Decision Tree Construction
- Avoiding Overfitting
- Minimum Description Length
- Handling Continuous valued attributes

Decision Tree

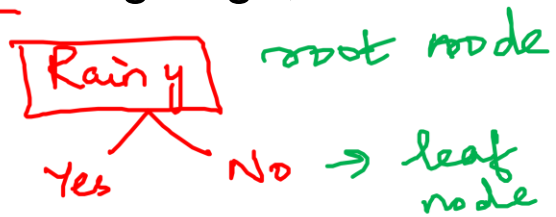
- The decision tree is a hierarchical tree-based algorithm that is used to classify or predict outcomes based on a set of rules. It works by splitting the data into subsets based on the values of the input features.
- The Decision tree algorithm works by selecting the best feature to split the data at each node.
- It has a tree-like structure that starts with one main question called the **root node** which represents the entire dataset. From there, the tree branches out into different possibilities based on features in the data.


How Decision Trees Work?
Aadhaar
Yes
Indian
?
Yes
Gas cylinder subsidy
✓
internal node
No
Leaf Node
Reaches final decision
Root Node
Starts with a question.
Branching
Yes/No questions split data.
class label

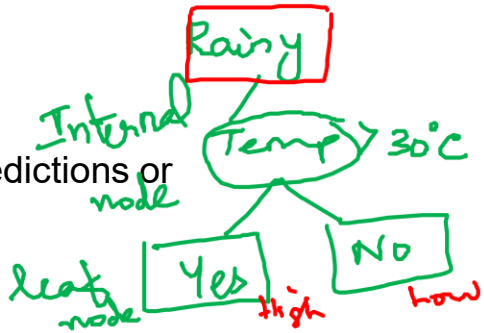
Building a **Decision tree**

New data : Temp = 28°C , Rain or not ?

- **Root Node:** Has no incoming edges, Zero or more outgoing edges
- **Internal Nodes:** Each has exactly one incoming edge, two or more outgoing edges
- **Leaf (Terminal) Nodes:** Each has exactly one incoming edge, No outgoing edges



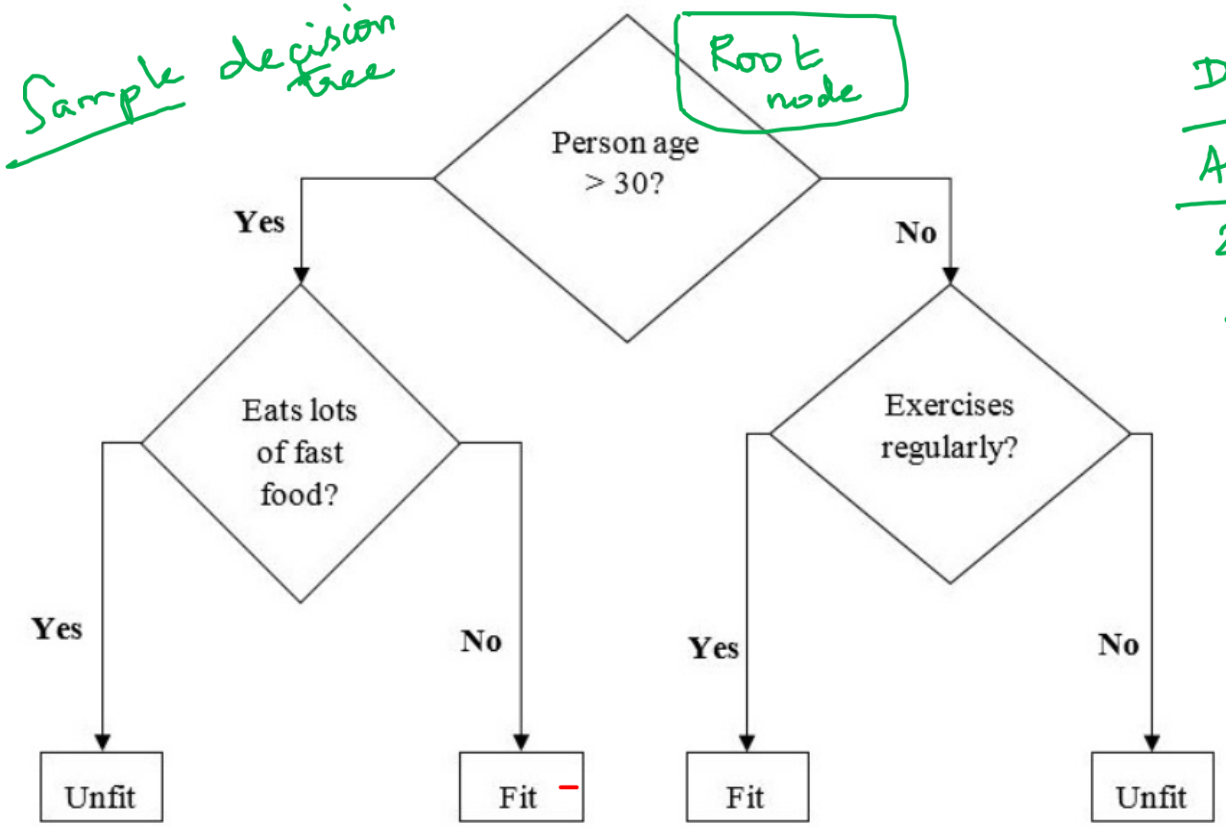
- Each leaf node assigned a class label
- Each non-leaf node contains a test condition on one of the attributes



- The resulting tree is a set of decision rules that can be used to make predictions or classify new data.

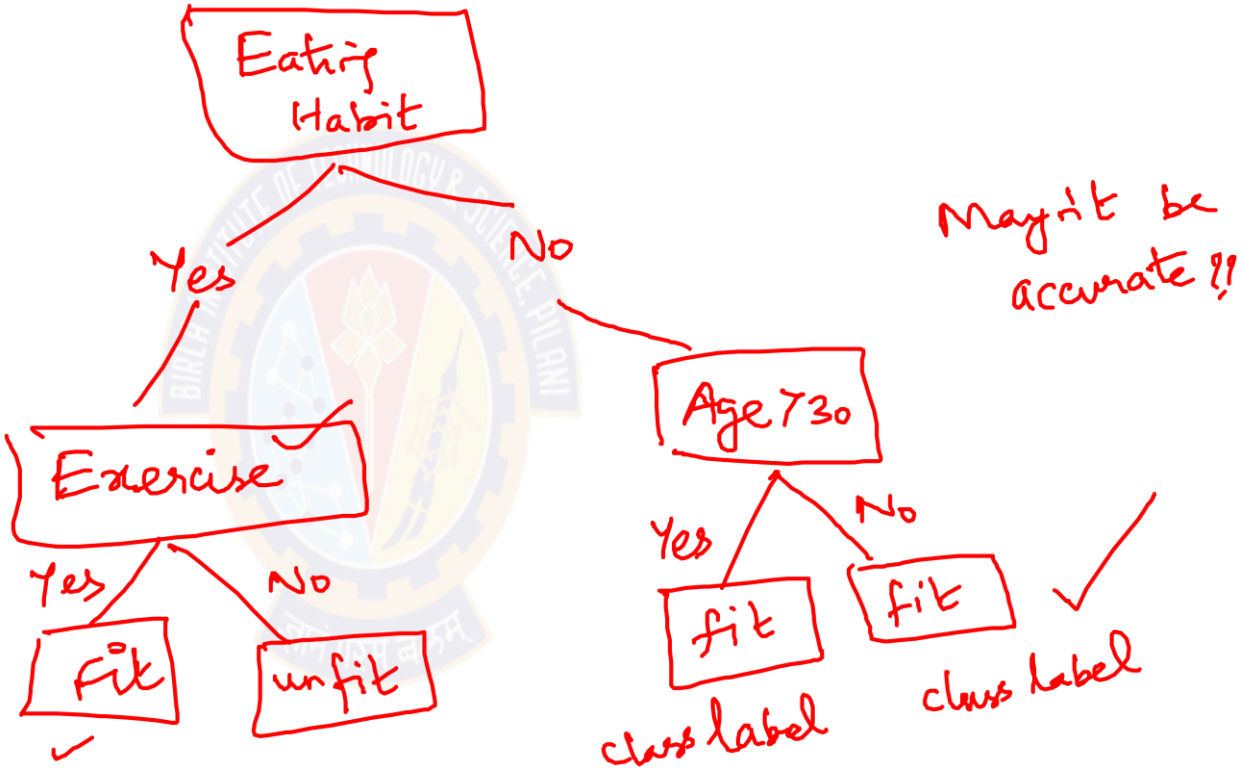
Example:

A binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits, is given below –



Data

Age	Eating	Exercise
25	Yes	Yes
28	Yes	No
31	No	Yes
32	No	No
29	No	Yes


Eating Habit
Yes
No
Exercise
Age > 30
Yes
No
Yes
No
fit
unfit
fit
fit
class label
class label
May it be accurate ??

Types of Decision Tree Algorithm

There are two main types of Decision Tree algorithm −

- **Classification Tree** − A classification tree is used to classify data into different classes or categories. It works by splitting the data into subsets based on the values of the input features and assigning each subset to a different class.

- **Regression Tree** − A regression tree is used to predict numerical values or continuous variables. It works by splitting the data into subsets based on the values of the input features and assigning each subset a numerical value.

Decision Tree Construction - Hunt's Algorithm

A decision tree is grown **recursively** by partitioning training records into **purer subsets**.

Step 1:

- If all records at a node belong to the same class → make the node a **leaf** labeled with that class.

Step 2:

- If records belong to more than one class → choose the **best attribute test condition** to split them.
- Create child nodes for each outcome, and distribute records.
- Recursively apply Hunt's algorithm to each child.

Decision Tree

Example: Deciding Whether to Play Tennis

- We want to decide whether it is a good day to play tennis.
- One approach is to pose a series of questions about the weather conditions.

First Question: What is the Outlook?

- If **Overcast** → **Yes (play tennis)**
- If **Sunny** → **Ask Humidity**
  - High → No (do not play)
  - Normal → Yes (play)
- If **Rain** → **Ask Wind**
  - Strong → No (do not play)
  - Weak → Yes (play)

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Key Idea:
By asking this sequence of questions, the decision tree helps us classify any new day into **Play Tennis = Yes or No**.

- Ask a series of **questions** about the **attributes** of the test record.
- Each answer leads to a follow-up question until we reach a decision.
- These questions and answers are organized into a **decision tree**, a hierarchical structure with nodes and directed edges.

Example:

Suppose you have:

•6 records with "Yes"

•4 records with "No"

You choose "Outlook" as the best splitting attribute. Then:

•Create child nodes for **Sunny**, **Overcast**, **Rain**.

•Send each record to its corresponding branch.

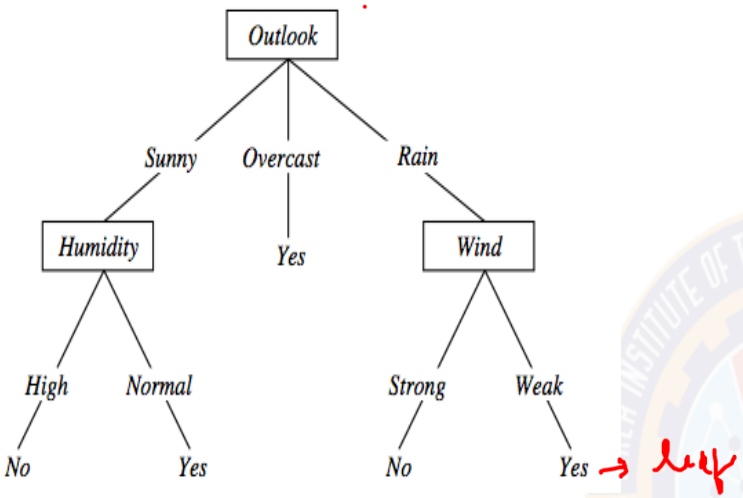
•Repeat Hunt's algorithm at each branch.

Recursion Continues Until:

•All data in a node belongs to one class (**pure**), or

•There are no more attributes left to split on.

Decision Theory: Interpretation



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- **Root Node:** Outlook
- **Internal Nodes:** Humidity, Wind
- **Leaf (Terminal) Nodes:** Yes, No
  - Each leaf node assigned a class label
  - Each non-leaf node contains a test condition on one of the attributes

Decision Trees – Representation & Expressiveness

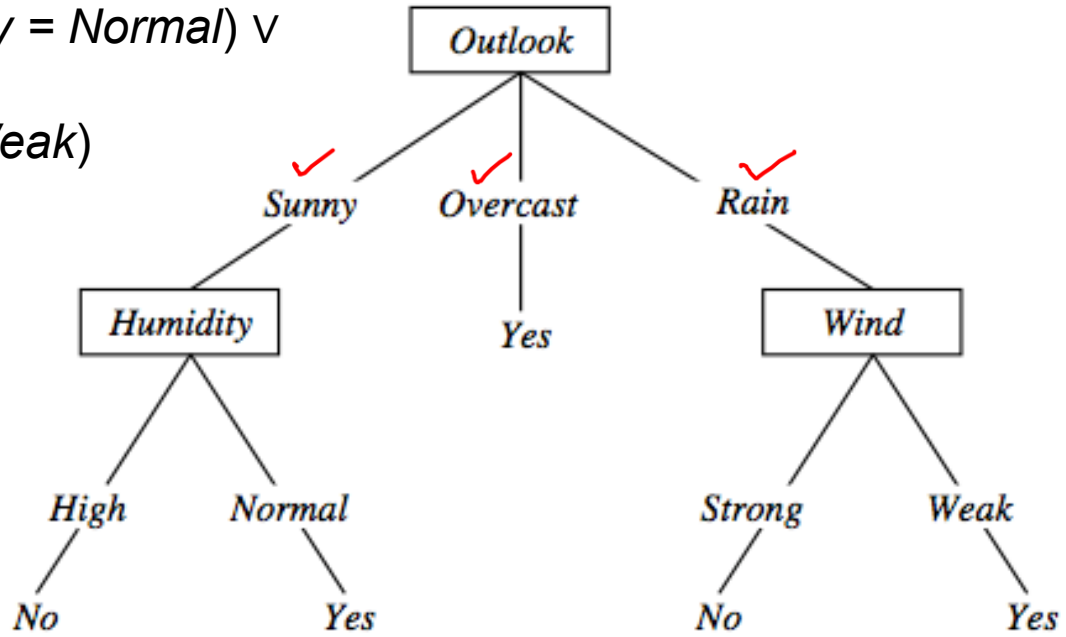
The entire tree is:

(Path1) *OR (Path2) OR (Path3)...*

This is called a **Disjunctive Normal Form (DNF)** — which is a **logical OR of ANDs**.

Decision trees represent a disjunction of conjunctions on constraints on the value of attributes:

(*Outlook* = *Sunny* ∧ *Humidity* = *Normal*) ∨
(*Outlook* = *Overcast*) ∨
(*Outlook* = *Rain* ∧ *Wind* = *Weak*)



⟨*Outlook=Sunny, Humidity=High, Wind=Strong*⟩ **No**

Why This Is Important

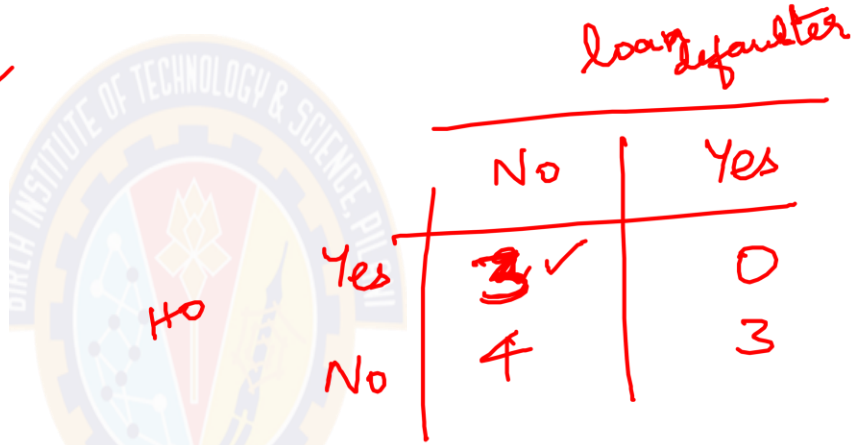
- Helps **understand and interpret** the model logically.
- You can **convert decision trees to rule-based systems** or **logical formulas**.
- Useful in **expert systems** and **explainable AI**.

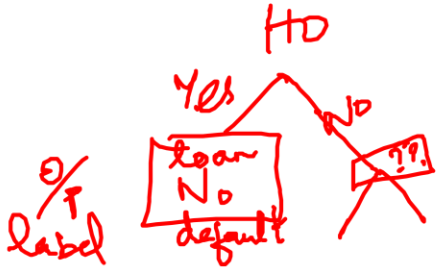
Decision Tree Construction: Example

Given a person's home ownership status, marital status, and annual income, will they default on a loan?

binary, categorical, continuous, class (label)

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes





Method 1:

Attribute	Type	Used for asking questions like:
Home Owner	Binary	"Is the person a homeowner?"
Marital Status	Categorical	"Is the person single, married, or divorced?"
Annual Income	Continuous	"Is the person's annual income less than 100K?" *(or other thresholds)*
Defaulted Borrower	Class label	(This is the **answer** or **target** you want to predict)

So, the decision tree will be built by asking questions like:

1."Is the person a homeowner?"

1. Yes → go one direction
2. No → go another

2.Then maybe: **"Is their income less than 100K?"**

3.Then maybe: **"What is their marital status?"**

And so on...

Method 2:

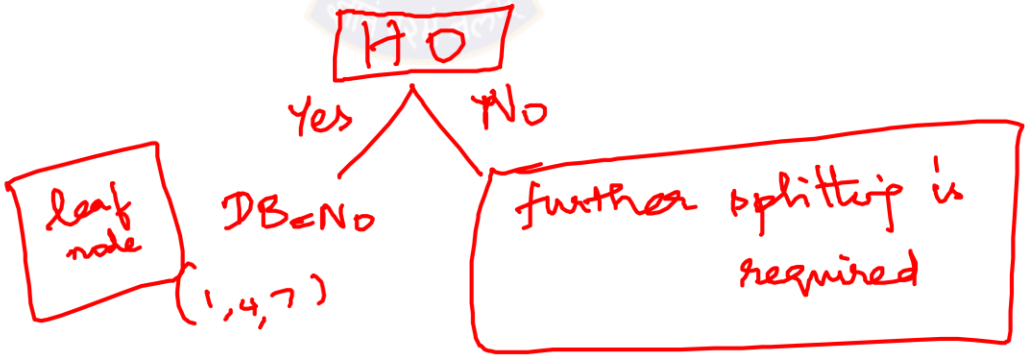
Step 1: Analyze the Target (Defaulted Borrower)

- **Yes**: Records 5, 8, 10
- **No**: Records 1, 2, 3, 4, 6, 7, 9
- Total: 3 Yes, 7 No

So majority is **No**, but let's break it down manually to look for **patterns**.

Step 2: Check `Home Owner`

Home Owner	Defaulted = Yes	Defaulted = No
Yes	0	1, 4, 7
No	5, 8, 10	2, 3, 6, 9



Step 3: Split Non-Homeowners (6 records: 2, 3, 5, 6, 8, 9, 10)

Tid	Marital Status	Income	Defaulted
2	Married	100K	No
3	Single	70K	No
5	Divorced	95K	Yes
6	Married	60K	No
8	Single	85K	Yes
9	Married	75K	No
10	Single	90K	Yes

Step 4: Try Splitting on `Marital Status`

Marital Status	Defaulted = Yes	Defaulted = No
Single	8, 10	3
Married	—	2, 6, 9
Divorced	5	—

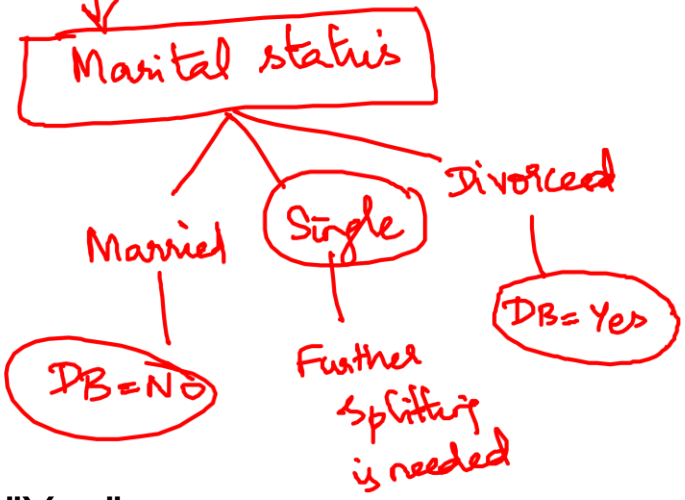
Marital status:

All married → No

Most single → Yes

Divorced → Yes

If HD = NO



let's split **Single** further.

Single people (3 records):

continuous variable

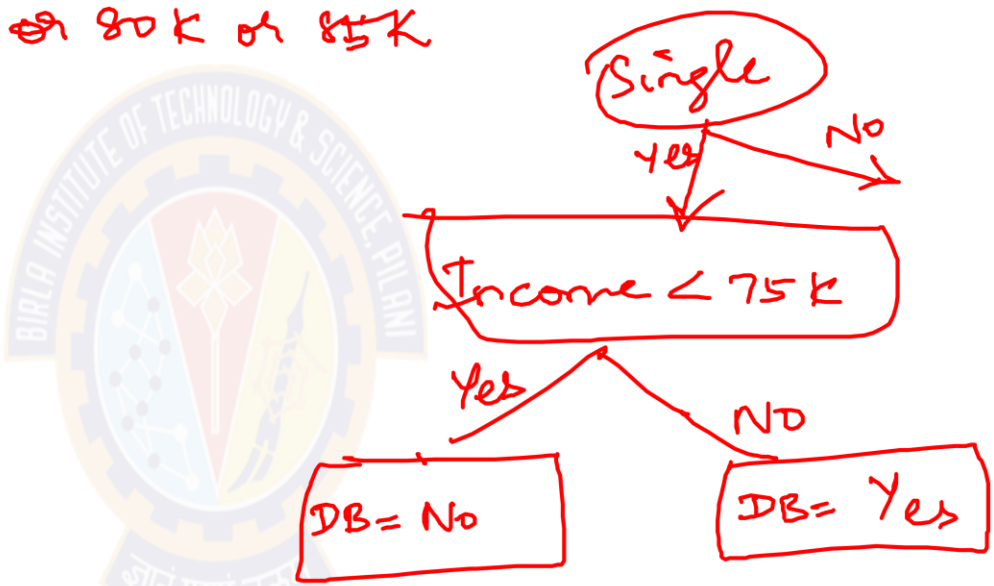
Tid	Income	Defaulted
3	70K	No
8	85K	Yes
10	90K	Yes

This is almost pure. Only Tid 3 is "No", rest are "Yes".

Let's split by **income threshold**. Try:

- **Is income < 75K?**
  - 70K → No
  - ≥ 85K → Yes

Works!



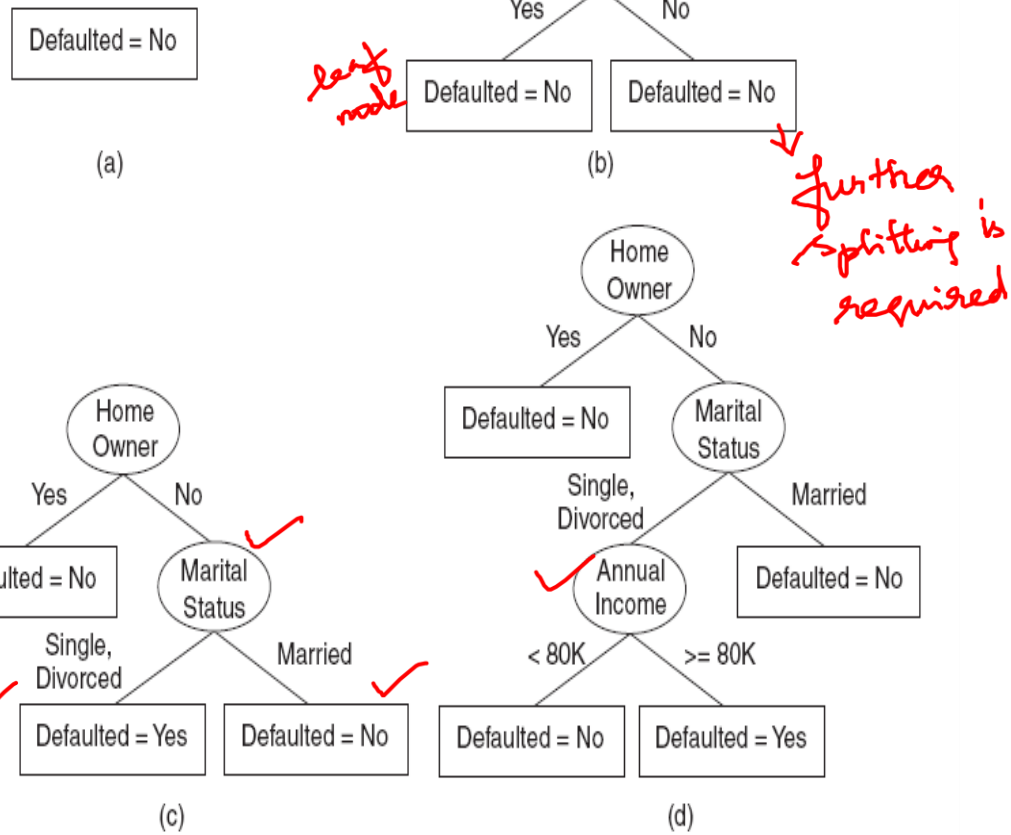
Decision Tree Construction: Example

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

binary, categorical, continuous, class

No Single 90K Yes

most cases



Design Issues in Decision Tree Induction

How to Split the Records?

- At each recursive step, choose an **attribute test condition**.
  - Define test conditions for different **attribute types** (categorical, continuous).

When to Stop Splitting?

- Stopping criteria are needed to prevent infinite growth.
  - All records at a node belong to the same class.
  - All records have identical attribute values.
  - Other criteria (e.g., maximum depth, minimum records per node) to stop earlier and avoid overfitting.

Methods for Expressing Attribute Test Conditions for different attribute type

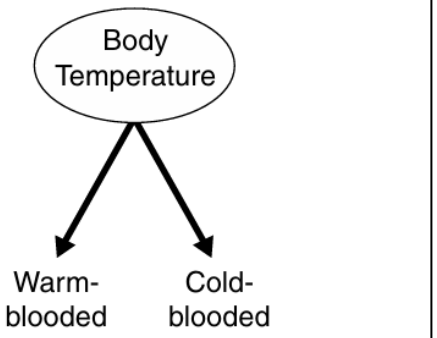
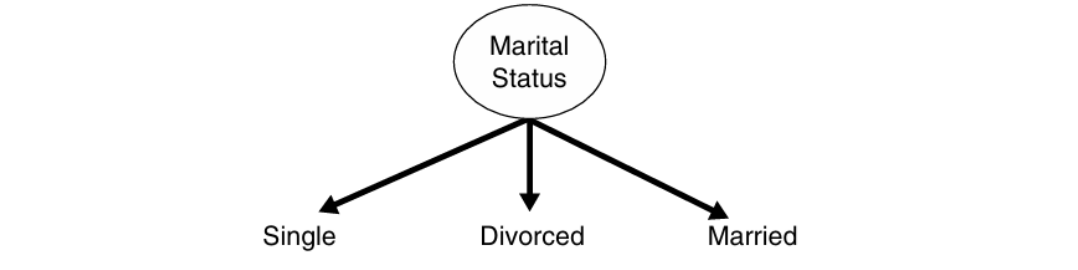
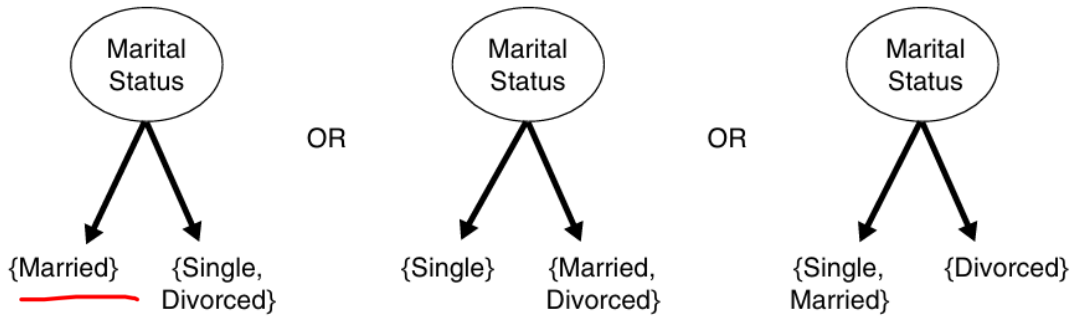


Figure 4.8. Test condition for binary attributes.



(a) Multiway split



(b) Binary split {by grouping attribute values}

Figure 4.9. Test conditions for nominal attributes.

Methods for Expressing Attribute Test Conditions

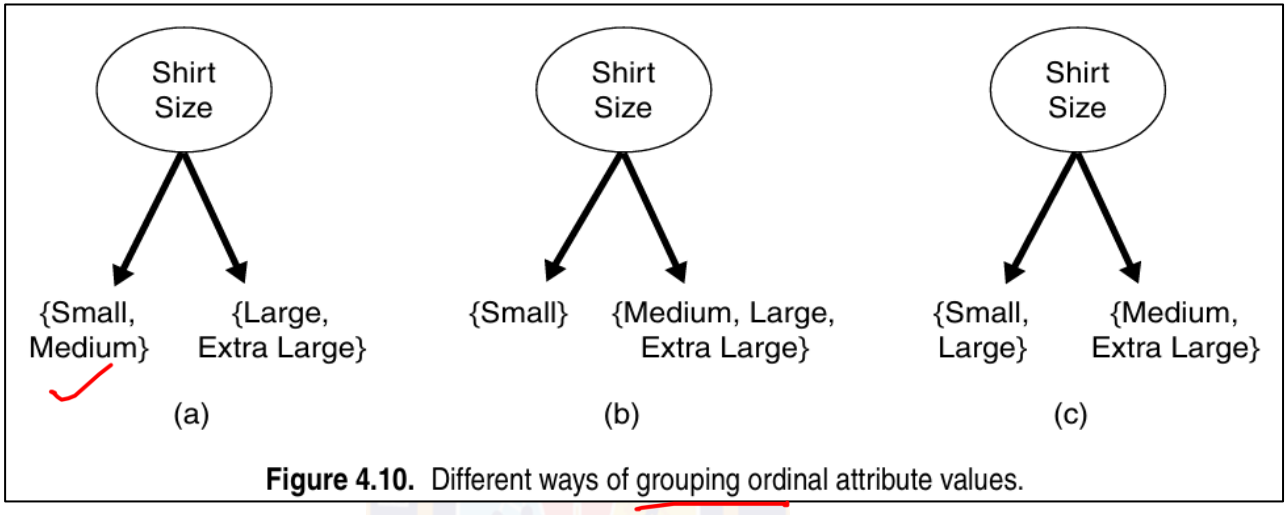


Figure 4.10. Different ways of grouping ordinal attribute values.

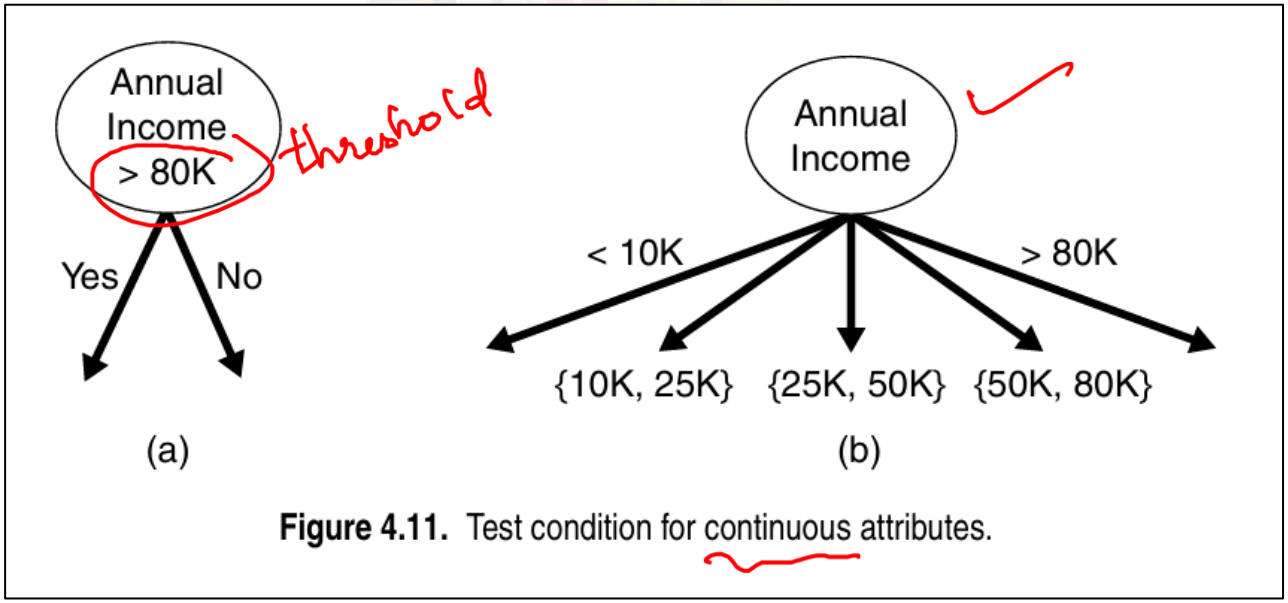


Figure 4.11. Test condition for continuous attributes.

Measures for Selecting the Best Split

- **Impurity** measures how **mixed** the class labels are in a node.

A node is:

- **Pure** if all records belong to the same class → impurity = 0.
- **Impure** if records are evenly split between classes → impurity is highest.

Before Splitting: 10 records of class 0, 10 records of class 1

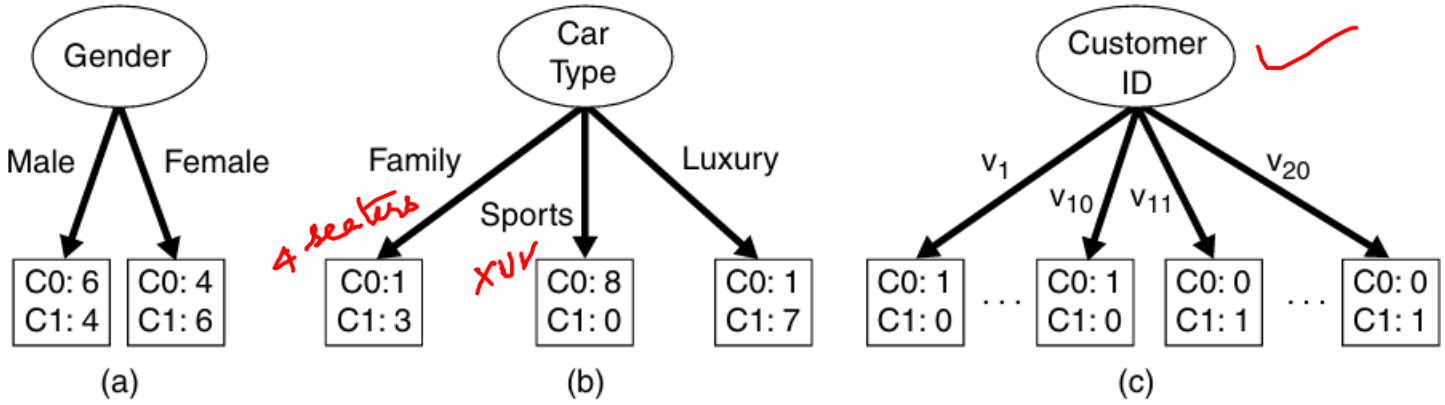


Figure 4.12. Multiway versus binary splits.

Which test condition is the best?

How to Measure Impurity?

- Entropy
- Gini Index
- Classification Error

Entropy

- **Entropy** is a measure of **impurity (uncertainty)** in a dataset (node).
- It quantifies the **amount of disorder** or **randomness** in the class distribution of records at a node.

$$Entropy(t) = -\sum_{i=1}^{C} p(i|t)\ \log_2 p(i|t)$$

Where:

- t = a node in the decision tree
- C = total number of classes
- i = class index (1, 2, ..., C)
- $p(i|t)$ = fraction (probability) of records at node t that belong to class i
- $\log_2$ = base-2 logarithm (measured in **bits**)

$$E = -\sum_{i=1}^{C} p_i \log_2 p_i$$

- **Entropy = 0** → node is **pure** (all records same class).
- **Entropy is maximum** when classes are evenly distributed

Entropy in binary Classification

- Entropy measures the *impurity* of a collection of examples. It depends from the distribution of the random variable *p.*
  - *S* is a collection of training examples
  - p_+ the proportion of positive examples in *S*
  - p_- the proportion of negative examples in *S*

$$\textit{Entropy}\ (S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Node N_1	Count
Class=0	0
Class=1	6

$$\text{Entropy} = -(0/6)\log_2(0/6) - (6/6)\log_2(6/6) = 0$$

Node N_2	Count
Class=0	1
Class=1	5

$$\text{Entropy} = -(1/6)\log_2(1/6) - (5/6)\log_2(5/6) = 0.650$$

Node N_3	Count
Class=0	3
Class=1	3

$$\text{Entropy} = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$$

Note: the log of a number < 1 is negative, $0 \le p \le 1$, $0 \le \textit{entropy} \le 1$

$$\log_2(1/6) = \frac{\log_{10}(1/6)}{\log(2)} = \frac{-0.7782}{0.3010}$$

Higher version of scientific calculator

$$\log_2(1/6) = \frac{\ln(1/6)}{\ln(2)} = -2.585$$

$$\log_2(5/6) = -0.2630$$

$$5/6 = 0.8333$$

Information gain as entropy reduction

- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- The higher the information gain the more effective the attribute in classifying training data.
- Expected reduction in entropy knowing *A*

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

S → the set of training examples

A → the attribute being tested

Values(A) → possible values of attribute A

S_v subset of *S* for which *A* has value *v*

Maximizing the gain = minimizing the weighted average impurity measure of children nodes

Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example: Information gain

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Parent node $S = [9^+, 5^-]$

$$Entropy(S) = -\left(\frac{9}{14}\log_2\frac{9}{14} + \frac{5}{14}\log_2\frac{5}{14}\right)$$
$$= -(0.6429 \cdot \log_2 0.6429 + 0.3571 \cdot \log_2 0.3571)$$
$$= 0.9403\ (\approx 0.94)$$

Child nodes after splitting on Wind

$S_{\text{Weak}} = [6^+, 2^-]$:

$$Entropy(S_{\text{Weak}}) = -\left(\frac{6}{8}\log_2\frac{6}{8} + \frac{2}{8}\log_2\frac{2}{8}\right) = 0.8113\ (\approx 0.811)$$

$S_{\text{Strong}} = [3^+, 3^-]$:

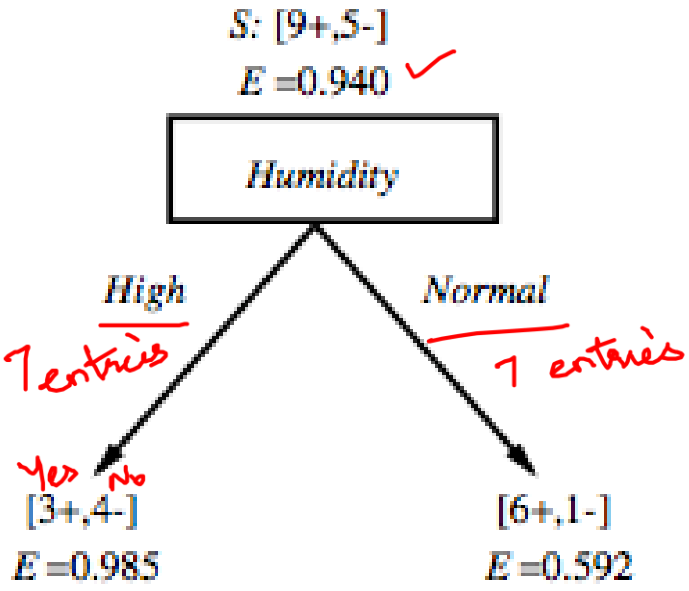
$$Entropy(S_{\text{Strong}}) = -\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) = 1.0000\ (= 1.00)$$

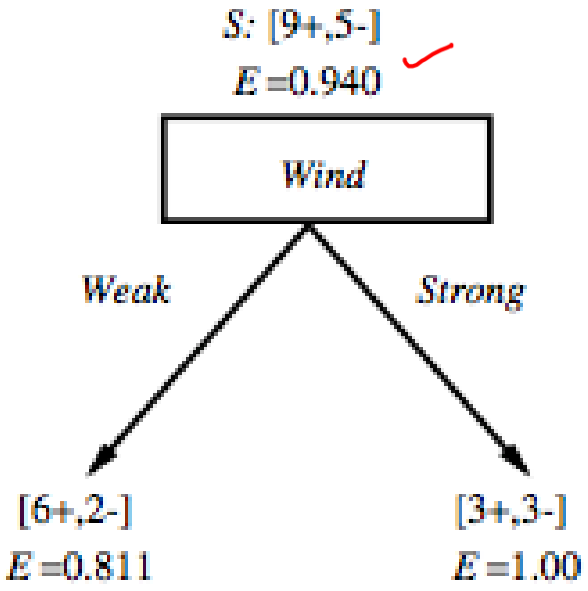
Information Gain for Wind

$$Gain(S, \text{Wind}) = Entropy(S) - \frac{8}{14}Entropy(S_{\text{Weak}}) - \frac{6}{14}Entropy(S_{\text{Strong}})$$
$$= 0.9403 - \frac{8}{14} \cdot 0.8113 - \frac{6}{14} \cdot 1.00$$
$$= 0.0481\ (\approx \mathbf{0.048})$$

Which attribute is the best classifier?

Which attribute is the best classifier?





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

First step: which attribute to test at the root?

Entropy of the whole dataset S

We have **14 instances**.

- PlayTennis = Yes → 9
- PlayTennis = No → 5

$$Entropy(S) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right)$$

$$Entropy(S) = -\frac{9}{14}\cdot 0.485 - \frac{5}{14}\cdot 0.737$$

$$Entropy(S) = 0.940$$

Gain(S, Outlook)

Outlook has 3 values: Sunny, Overcast, Rain.

- **Sunny (5 instances)** → 2 Yes, 3 No

$$Entropy(Sunny) = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.971$$

- **Overcast (4 instances)** → 4 Yes, 0 No

$$Entropy(Overcast) = 0$$

- **Rain (5 instances)** → 3 Yes, 2 No

$$Entropy(Rain) = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.971$$

Weighted entropy:

$$Entropy_{Outlook} = \frac{5}{14}(0.971) + \frac{4}{14}(0) + \frac{5}{14}(0.971) = 0.693$$

$$Gain(S, Outlook) = 0.940 - 0.693 = 0.247$$

First step: which attribute to test at the root?

Gain(S, Humidity)

Humidity has 2 values: High, Normal.

- **High (7 instances)** → 3 Yes, 4 No

$$Entropy(High) = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.985$$

- **Normal (7 instances)** → 6 Yes, 1 No

$$Entropy(Normal) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.592$$

Weighted entropy:

$$Entropy_{Humidity} = \frac{7}{14}(0.985) + \frac{7}{14}(0.592) = 0.789$$

$$Gain(S, Humidity) = 0.940 - 0.789 = 0.151$$

Repeated

Gain(S, Wind)

Wind has 2 values: Weak, Strong.

- **Weak (8 instances)** → 6 Yes, 2 No

$$Entropy(Weak) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.811$$

- **Strong (6 instances)** → 3 Yes, 3 No

$$Entropy(Strong) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1.0$$

Weighted entropy:

$$Entropy_{Wind} = \frac{8}{14}(0.811) + \frac{6}{14}(1.0) = 0.892$$

$$Gain(S, Wind) = 0.940 - 0.892 = 0.048$$

First step: which attribute to test at the root?

Gain(S, Temperature)

Temperature has 3 values: Hot, Mild, Cool.

- **Hot (4 instances)** → 2 Yes, 2 No

prob(Hot) = 4/14

$$Entropy(Hot) = 1.0$$

- **Mild (6 instances)** → 4 Yes, 2 No

prob(mild) = 6/14

$$Entropy(Mild) = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.918$$

- **Cool (4 instances)** → 3 Yes, 1 No

prob(Cool) = 4/14

$$Entropy(Cool) = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.811$$

Weighted entropy:

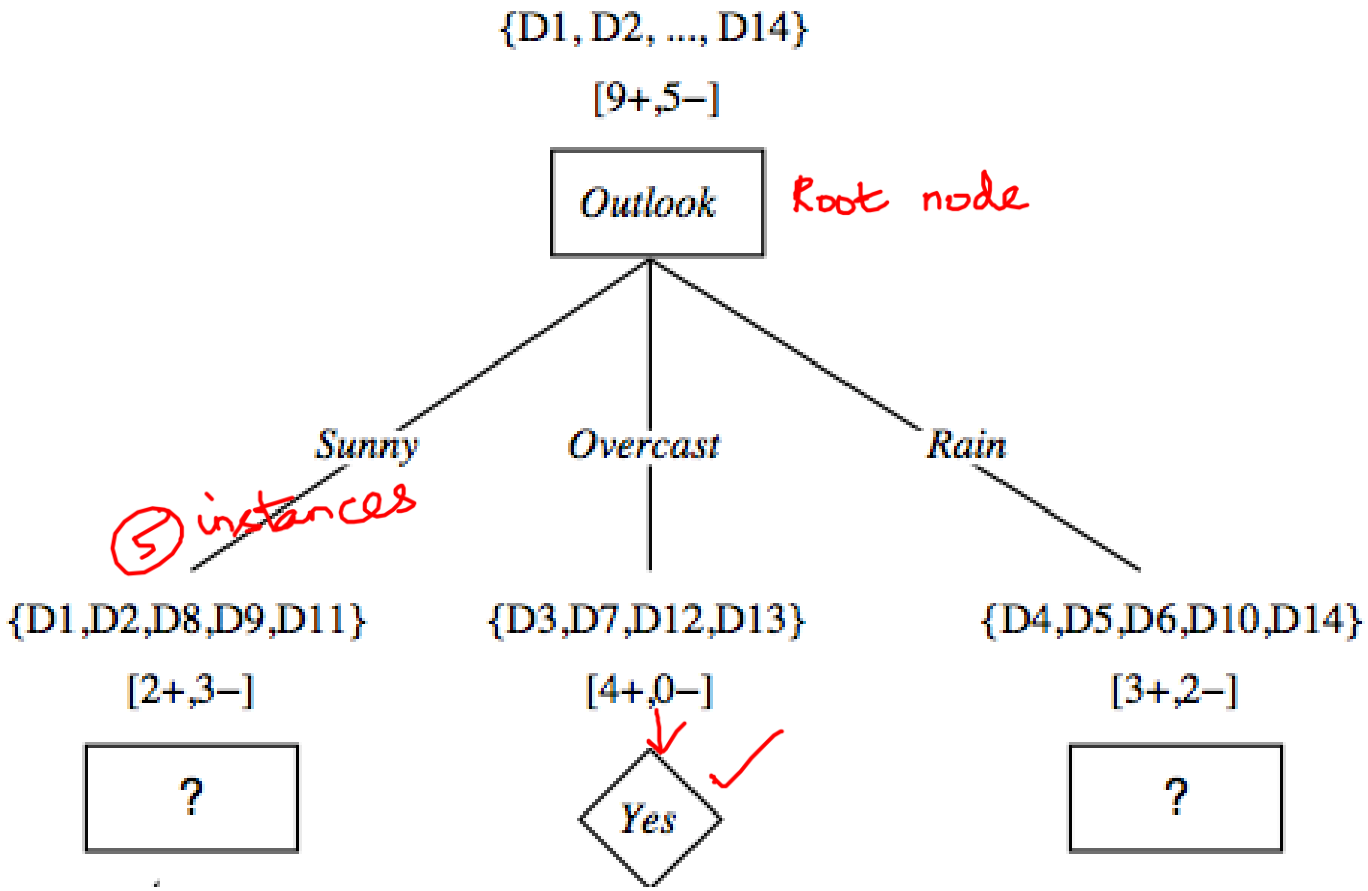
$$Entropy_{Temp} = \frac{4}{14}(1.0) + \frac{6}{14}(0.918) + \frac{4}{14}(0.811) = 0.911$$

$$Gain(S, Temperature) = 0.940 - 0.911 = 0.029$$

First step: which attribute to test at the root?

- Which attribute should be tested at the root?
  - *Gain*(*S*, *Outlook*) = **0.247**
  - *Gain*(*S*, *Humidity*) = 0.151
  - *Gain*(*S*, *Wind*) = 0.048
  - *Gain*(*S*, *Temperature*) = 0.029
- ***Outlook*** provides the best prediction for the target
- Lets, grow the tree:
  - add to the tree a successor for each possible value of *Outlook*
  - partition the training samples according to the value of *Outlook*

After first step



Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Second step

- Working on *Outlook*=*Sunny* node:

  $Gain(S_{Sunny}, Humidity) = 0.971 - 3/5 \times 0.0 - 2/5 \times 0.0 = \mathbf{0.971}$

  $Gain(S_{Sunny}, Wind) = 0.971 - 2/5 \times 1.0 - 3/5 \times 0.918 = 0\ .020$

  $Gain(S_{Sunny}, Temp.) = 0.971 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 =$ 0.571

- ***Humidity*** provides the best prediction for the target
- Lets, grow the tree:
  - add to the tree a successor for each possible value of *Humidity*
  - partition the training samples according to the value of *Humidity*

Third step

Branch on Outlook = Rain

Day	Temperature	Humidity	Wind	PlayTennis
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Total = 5 instances

- **Yes = 3**
- **No = 2**

Entropy ≈ **0.971** (as we calculated earlier)

So, this subset is **not pure**, and we need to **split further**.

$$p_+ = \frac{3}{5} = 0.6$$
$$p_- = \frac{2}{5} = 0.4$$

$$\text{Entropy}_{\text{Rain}} = -(0.6 \log_2(0.6)) - (0.4 \log_2(0.4))$$
$$= -(0.6 \cdot -0.737) - (0.4 \cdot -1.322) = 0.4422 + 0.5288 = \boxed{0.971}$$

1. Wind

Values: Weak, Strong

- Weak: D4, D5, D10 → Yes, Yes, Yes (3 instances, all Yes)
- Strong: D6, D14 → No, No (2 instances, all No)

Entropies:

- Weak → $E = 0$
- Strong → $E = 0$

Weighted entropy:

$$\text{Entropy}(\text{Wind}) = \frac{3}{5} \cdot 0 + \frac{2}{5} \cdot 0 = 0$$

Information Gain:

$$IG(Wind) = 0.971 - 0 = \boxed{0.971}$$

2. Humidity (5 instances)

Values: High, Normal

- **High**: D4 (Yes), D14 (No) → 1 Yes, 1 No
- **Normal**: D5, D6, D10 → Yes, No, Yes → 2 Yes, 1 No

Entropy for High:

$$p_+ = 0.5,\ p_- = 0.5$$

$$E(High) = -0.5\log_2(0.5) - 0.5\log_2(0.5) = 1$$

Entropy for Normal:

$$p_+ = \frac{2}{3},\ p_- = \frac{1}{3}$$

$$E(Normal) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) = -\left(\frac{2}{3}\cdot -0.585 + \frac{1}{3}\cdot -1.585\right) = 0.918$$

Weighted entropy:

$$\text{Entropy}(\text{Humidity}) = \frac{2}{5}(1) + \frac{3}{5}(0.918) = 0.4 + 0.5508 = \boxed{0.9508}$$

Information Gain:

$$IG(Humidity) = 0.971 - 0.9508 = \boxed{0.0202}$$

3. Temperature

Values: Mild, Cool

- **Mild**: D4 (Yes), D10 (Yes), D14 (No) → 2 Yes, 1 No
- **Cool**: D5 (Yes), D6 (No) → 1 Yes, 1 No

Entropy for Mild:

$$p_+ = \frac{2}{3},\ p_- = \frac{1}{3} \Rightarrow E = 0.918$$

Entropy for Cool:

$$p_+ = 0.5,\ p_- = 0.5 \Rightarrow E = 1$$

Weighted entropy:

$$\text{Entropy(Temp)} = \frac{3}{5}(0.918) + \frac{2}{5}(1) = 0.5508 + 0.4 = \boxed{0.9508}$$

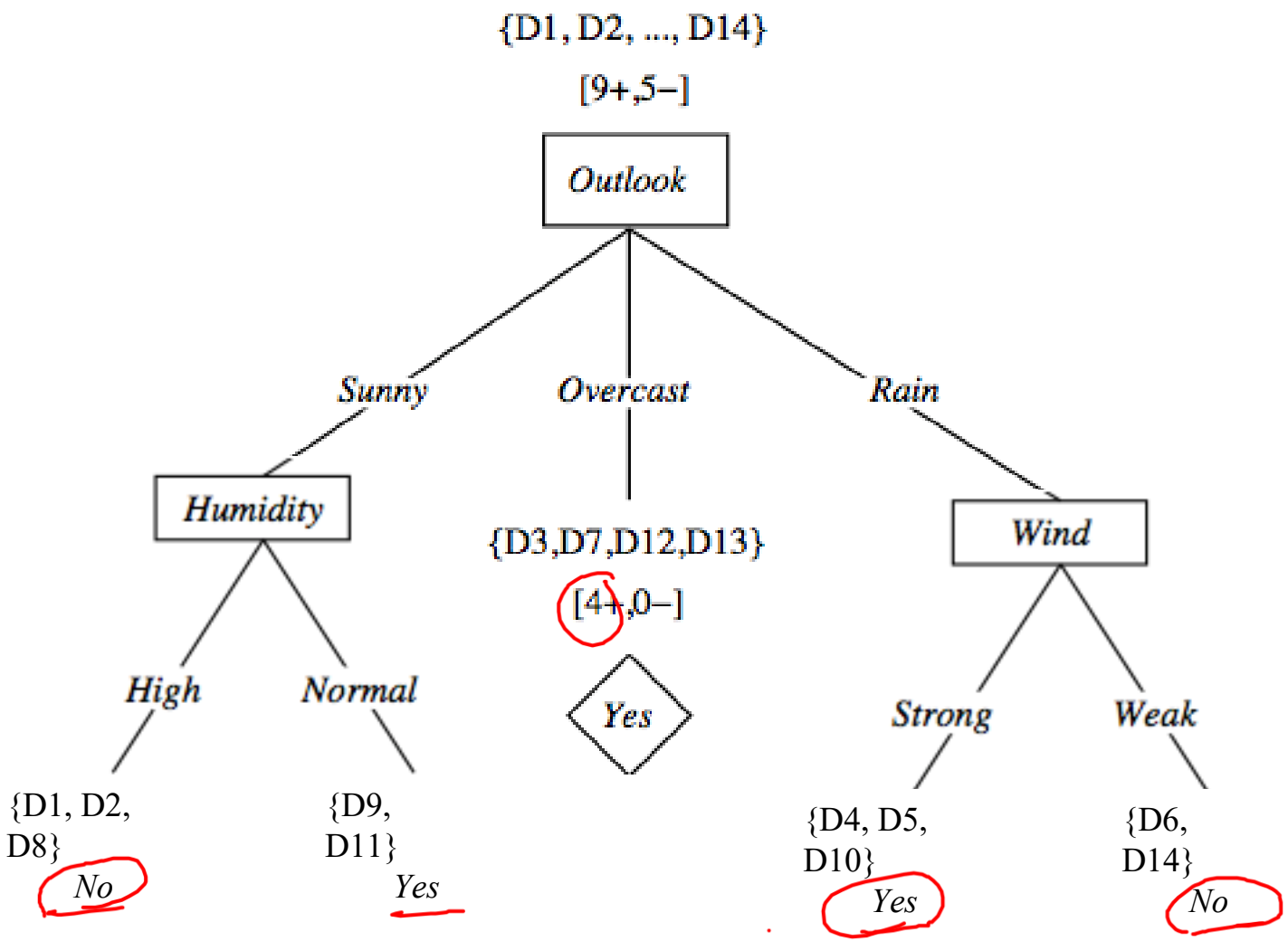
Information Gain:

$$IG(Temperature) = 0.971 - 0.9508 = \boxed{0.0202}$$

Final result:

Attribute	Information Gain
Wind	**0.971**
Humidity	0.0202
Temperature	0.0202

Second and third steps



ID3: algorithm -Summary

ID3(*X, T, Attrs*) *X*: training examples:
T: target attribute (e.g. *PlayTennis*),

Attrs: other attributes, initially all attributes

```
Create Root node
If all X's are +, return Root with class +
If all X's are −, return Root with class −
If Attrs is empty return Root with class most common value of T in X
else
A ← best attribute; decision attribute for Root ← A
For each possible value v_i of A:
  - add a new branch below Root, for test A = v_i
  - X_i ← subset of X with A = v_i
  - If X_i is empty then add a new leaf with class the most common value of T in X
      else add the subtree generated by ID3(X_i, T, Attrs − {A})
return Root
```

ID3 → Iterative Dichotomiser 3

fundamental algorithm for building decision tree

1986

Ross Quinlan → top-down, greedy approach

ID3 repeatedly selects the **most informative attribute**, splits the dataset, and builds a **tree structure that classifies the data.**

The algorithm **recursively builds the tree** until:

- All records in a node belong to the **same class**, or
- **No attributes remain**, or
- The subset becomes **empty**.

Decision trees

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Can easily handle redundant or irrelevant attributes (unless the attributes are interacting)
- Disadvantages:
  - Space of possible decision trees is exponentially large.
  - -- Greedy approaches are often unable to find the best tree.
  - Does not take into account interactions between attributes
  - Each decision boundary involves only a single attribute

Fit/unfit

Male/Female

Decision Trees – Problem Type 1

Entropy of the Decision Feature EnjoySport:

Target Attribute (Decision Class)

- The column **EnjoySport?** is the class label (Yes/No).
- We have **4 examples**:
  - Yes → 3
  - No → 1

= **Entropy** $_{\text{DecisionClass}}$**(S) = Σ-p(c) log$_2$ p(c)**

= E(3,1) = – (3/4) $\log_2$ (3/4) – (1/4) $\log_2$ (1/4) = 0.4421+ 0.5 = 0.94

o/p label

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Note:

- Class balancing is recommended before building
- Entropy is always between 0 to log base 2 k, where k is the number of classes you have.

Decision Trees – Problem Type 1

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Entropy of the Decision Feature EnjoySport:

= **Entropy** $_{\text{DecisionClass}}$**(S) =** Σ**-p(c) log$_2$ p(c)**

= E(3,1) = – (3/4) log$_2$ (3/4) – (1/4) log$_2$ (1/4) = 0.4421+ 0.5 = 0.94

Sky	Enjoy	Not Enjoy
Sunny	3	0
Rainy	0	1

Entropy of the attribute Sky:

= p(Sunny) . E(3,0) + p(Rainy). E(0,1)

= (3/4) [– (3/3) log$_2$ (3/3) – (0/3) log$_2$ (0/3)] + (1/4) [– (0/1) log$_2$ (0/1) – (1/1) log$_2$ (1/1)]

= 0

Air Temp	Enjoy	Not Enjoy
Warm	3	0
Cold	0	1

Entropy of the attribute AirTemp:

= p(Warm) . E(3,0) + p(Cold). E(0,1)

= (3/4) [– (3/3) log$_2$ (3/3) – (0/3) log$_2$ (0/3)] + (1/4) [– (0/1) log$_2$ (0/1) – (1/1) log$_2$ (1/1)]

= 0

Decision Trees – Problem Type 1

Entropy of the Decision Feature EnjoySport:

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

= **Entropy** $_{\text{DecisionClass}}$**(S) =** Σ**-p(c) log**$_2$ **p(c)**

= E(3,1) = – (3/4) $\log_2$ (3/4) – (1/4) $\log_2$ (1/4) = 0.94

Weighted

Entropy of the attribute Humidity :

Humidity	Enjoy	Not Enjoy
Normal	1	0
High	2	1

= p(Normal) . E(1,0) + p(High). E(2,1)

= (1/4) [– (1/1) $\log_2$ (1/1) – (0/1) $\log_2$ (0/1)] + (3/4) [– (2/3) $\log_2$ (2/3) – (1/3) $\log_2$ (1/3)]

= 0 + ¾ [0.39 + 0.52] = 0.68

Entropy of the attribute Wind :

Wind	Enjoy	Not Enjoy
Strong	3	1

= p(Strong) . E(3,1)

= (4/4) [– (3/4) $\log_2$ (3/4) – (1/4) $\log_2$ (1/4)]

= 1(0.94) = 0.94

Decision Trees – Problem Type 1

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Entropy of the Decision Feature EnjoySport:

= **Entropy** $_{\text{DecisionClass}}$**(S) =** Σ**-p(c) log**$_2$ **p(c)**

= E(3,1) = – (3/4) $\log_2$ (3/4) – (1/4) $\log_2$ (1/4) = 0.94

Forecast	Enjoy	Not Enjoy
Same	2	0
Change	1	1

Entropy of the attribute Forecast :

= p(Same) . E(2,0) + p(Change). E(1,1)

= (2/4) [– (2/2) $\log_2$ (2/2) – (0/2) $\log_2$ (0/2)] + (2/4) [– (1/2) $\log_2$ (1/2) – (1/2) $\log_2$ (1/2)]

= 0 + 0.5 = 0.5

Decision Trees – Problem Type 1

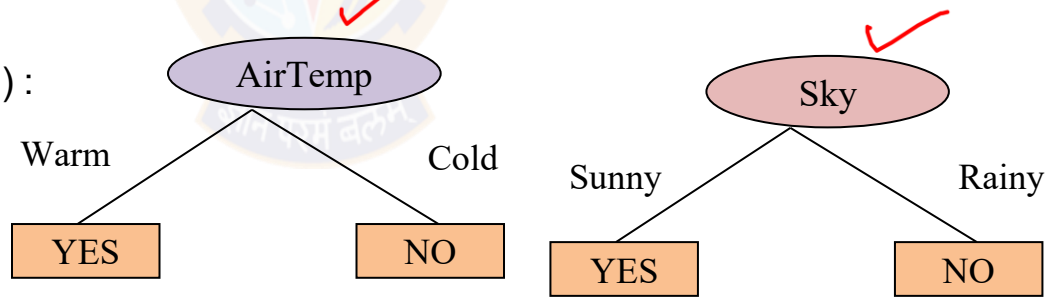
Entropy of the Decision Feature *EnjoySport*: **0.94**
Entropy of the attribute *Sky*: **0**
Entropy of the attribute *AirTemp*: **0**
Entropy of the attribute *Humidity*: **0.68**
Entropy of the attribute *Wind*: **0.94**
Entropy of the attribute *Forecast*: **0.5**

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Attribute that generalizes the model faster = That which reduces the randomness if used to decide = That which have the highest info gain = SKY & AIRTEMP

Decision Tree constructed

(both are equally correct decision tree) :



Decision Trees – Problem Type 2

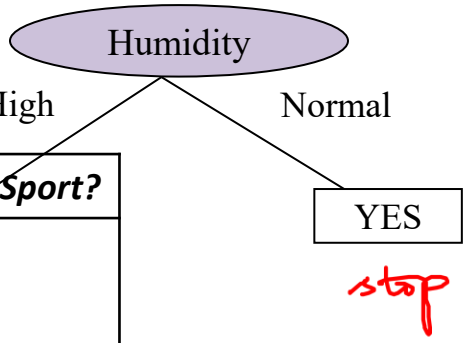
(If there are heterogeneous nodes, continue the decision tree construction)

Sub Problem which has more scope to Model: None

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes

Assumed sample data in the node

Sky	AirTemp	Wind	Forecast	*EnjoySport?*
Sunny	Warm	Strong	Same	Yes
Rainy	Cold	Strong	Change	No
Sunny	Warm	Strong	Change	Yes



Convergence Criteria :

Repeat the step again for the remaining features till:

- All training data are learnt
- No more learning is possible
- Constraints posed by the hyperparameter satisfied (More on this will be discussed in handling overfitting)

Convergence of Algorithm

- Conditions for stopping partitioning.
- All samples for a given node belong to the same class.
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf.
- There are no samples left.

Decision Trees – Problem Type 3

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	????

Sky has instances: Rainy (1), Sunny (3 instance)

New

Humidity: High → No, Normal → Yes

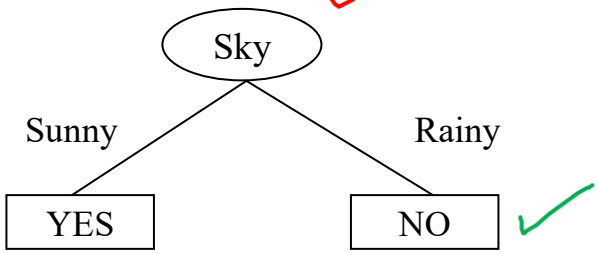
Entropy(Sky) = $-\left[\frac{3}{4}\log_2\left(\frac{3}{4}\right) + \frac{1}{4}\log_2\left(\frac{1}{4}\right)\right]$

Predict the class for new input instance

Follow the tree:

•Check **Sky** → it's **Rainy** → go to the right leaf → **NO**.

Predicted class: *EnjoySport? = NO*.



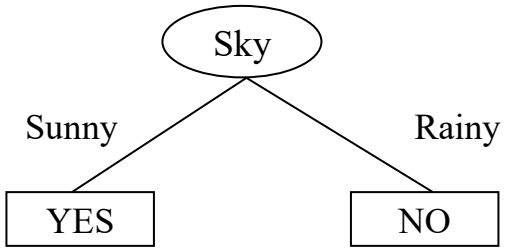
Problem Type 4 – Evaluate the model Confusion Matrix

- The evaluation discussed under module 4 is applicable here
- By default , model's predictive quality is evaluated **only by the unseen validation/test instances.**
- When training performance (eg., accuracy) is required for comparison, the training instances are used for assessment

Actual class\Predicted class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Negatives (FN)**
$\neg C_1$	**False Positives (FP)**	**True Negatives (TN)**

•$CM_{i,j}$ indicates the number of tuples of class i that were labelled by the classifier as class j.

Sky	AirTemp	Humidity	Wind	Forecast	EnjoySport ?
Sunny	Warm	Normal	Strong	Same	Yes
Sunny	Warm	High	Strong	Same	Yes
Rainy	Cold	High	Strong	Change	No
Sunny	Warm	High	Strong	Change	Yes
Rainy	Warm	Normal	Breeze	Same	Yes



The new row:

- "Rainy", "Warm", "Normal", "Breeze", "Same"
- None of the training rows exactly match this. But using a simple rule-based match from training data, if the model generalizes from the positive examples (majority being “Yes”), it **might predict "Yes"**.

Confusion Matrix:

	Predicted: Yes	**Predicted: No**
Actual: Yes	TP = 1	FN = 0
Actual: No	FP = 0	TN = 0

This confusion matrix is only based on **one test sample** (the bottom row), so it's very limited. To get meaningful evaluation, you'd need more test samples.

Practice Exercises (for Students)

1. For this similar problem discussed in class build a decision tree classifier using ID3 algorithm ie., Information gain and entropy measures. Grow the complete decision tree.
2. Can "Object ID" be used as one of key feature to construct the tree? Justify your answer on the results of entropy calculated.
3. Use the test data to construct the confusion matrix and find the precision , recall, F-score.
4. Compare the training accuracy vs test accuracy for the model built in part 1)

	Training Data			o/p
Object ID	**Shape**	**Colour**	**Size**	**Action**
OB101	Round	Green	Small	Reject
OB102	Square	Black	Big	Allow
OB103	Square	Brown	Big	Allow
OB104	Round	Brown	Small	Reject
OB105	Square	Green	Big	Allow
OB105	Square	Brown	Small	Reject
OB106	Oval	Green	Big	Reject
OB107	Oval	Brown	Small	Allow
OB108	Oval	Green	Small	Reject

	Test Data / Prune Set/ Validation Set			
Object ID	**Shape**	**Colour**	**Size**	**Action**
OB109	Oval	Black	Small	Reject
OB110	Round	Brown	Big	Allow
OB111	Square	Brown	Big	Allow
OB112	Oval	Green	Small	Allow

Answer - Hint

Final ID3 tree

- **Shape**
  - **Round → Reject**
  - **Square → Size**
    - **Big → Allow**
    - **Small → Reject**
  - **Oval → Colour**
    - **Brown → Allow**
    - **Green → Reject**

No, **Object ID cannot be used as a splitting feature** because it is just an identifier. Each ID is unique, so splitting on it will simply memorize the training data (entropy = 0) without learning any real pattern. Thi causes **overfitting** and the model won't generalize to new data.

Confusion matrix (Positive = Allow):

	Pred Allow	Pred Reject
True Allow	1 (TP)	2 (FN)
True Reject	0 (FP)	1 (TN)

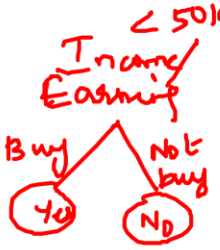
- **Training accuracy:** 100% (pure leaves after two levels).
- **Test accuracy: 50%.**

Interpretation: the tree fits the small training set perfectly and **overfits**—it doesn't generalize as well to new combinations (e.g., unseen "Oval & Black"). This also illustrates why **using Object ID would make overfitting worse** (near-100% training accuracy with poor/undefined test behavior for new IDs).

Alternative to Entropy: Gini Index

Classification And Regression Tree

- The **Gini Index** (used in decision trees like CART) measures the impurity of a dataset.



Formally, for a node t:

$$Gini(t) = 1 - \sum_{i=1}^{c} \left(p(i|t)\right)^2$$

where:

- c = number of classes
- $p(i|t)$ = proportion of samples in node t that belong to class i

$P_+ = 0/6$

$P_- = 6/6$

Node N_1	Count
Class=0	0
Class=1	6

+ve
-ve

Tot = 6

Gini $= 1 - (0/6)^2 - (6/6)^2 = 0$ Pure

Node N_2	Count
Class=0	1
Class=1	5

Tot = 6

Gini $= 1 - (1/6)^2 - (5/6)^2 = 0.278$

$Gini_{(N_2)} = 0.278$

Node N_3	Count
Class=0	3
Class=1	3

Gini $= 1 - (3/6)^2 - (3/6)^2 = 0.5$

	Parent
No	7
Yes	3
Gini = 0.420	

Tot = 10

$1 - \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.420.$

choose the one with the **lowest Gini index** — meaning the best purity (or least impurity) after the split.

Gini Index Formula

For a binary classification (e.g., Yes/No), Gini Impurity is:

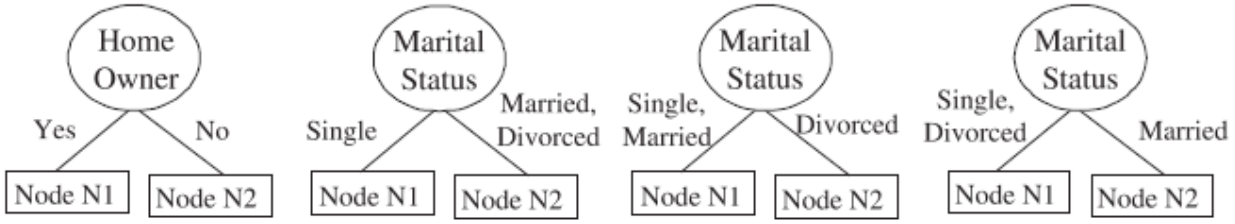
$$\text{Gini}(D) = 1 - p_{\text{Yes}}^2 - p_{\text{No}}^2$$

For a split into subsets D_1 and D_2, the **Gini Index** is:

$$\text{GiniIndex} = \frac{|D_1|}{|D|} \cdot \text{Gini}(D_1) + \frac{|D_2|}{|D|} \cdot \text{Gini}(D_2)$$

Splitting of Attributes :
Gini Index : Binary Splits

ID	Home Owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes
6	No	Single	60000	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes
9	No	Married	75000	No
10	No	Single	90000	Yes



Splitting of Attributes : Gini Index : Binary Splits

Parent Node Gini (before split)

From the full dataset of 10 records:

- **Yes (Defaulted)** = 3, **No (Defaulted)** = 7

$$Gini_{parent} = 1 - \left(\left(\frac{3}{10}\right)^2 + \left(\frac{7}{10}\right)^2\right)$$

$$= 1 - (0.09 + 0.49) = 1 - 0.58 = 0.420$$

ID	Home Owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes
6	No	Single	60000	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes
9	No	Married	75000	No
10	No	Single	90000	Yes

Splitting Attribute: HomeOwner

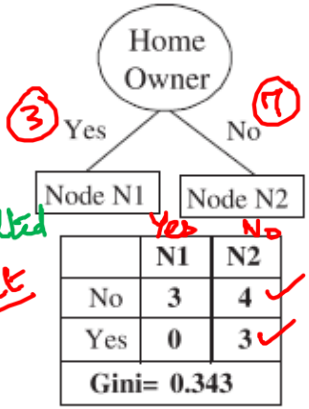
Gini index for the child nodes

N1 = 1- (0/3)^2 - (3/3)^2 = 0

N2 =1- (3/7)^2 - (4/7)^2 = 0.490

There are 3 'Yes' in HO, 7 'No' in HO

Root node ??



Defaulted output

Weighted Gini Index for children

$$I(\text{children}) = \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

$$(3/10) \times 0 + (7/10) \times 0.490 = 0.343,$$

$\frac{3}{10}$ Gini N_1 + $\frac{7}{10}$ Gini N_2

The gain using Home Owner as splitting attribute is

0.420 - 0.343 =0.077

$$Gain = Gini_{parent} - Gini_{children}$$

For Node N1

It is 'Yes' in HO and 'No' in the defaulted

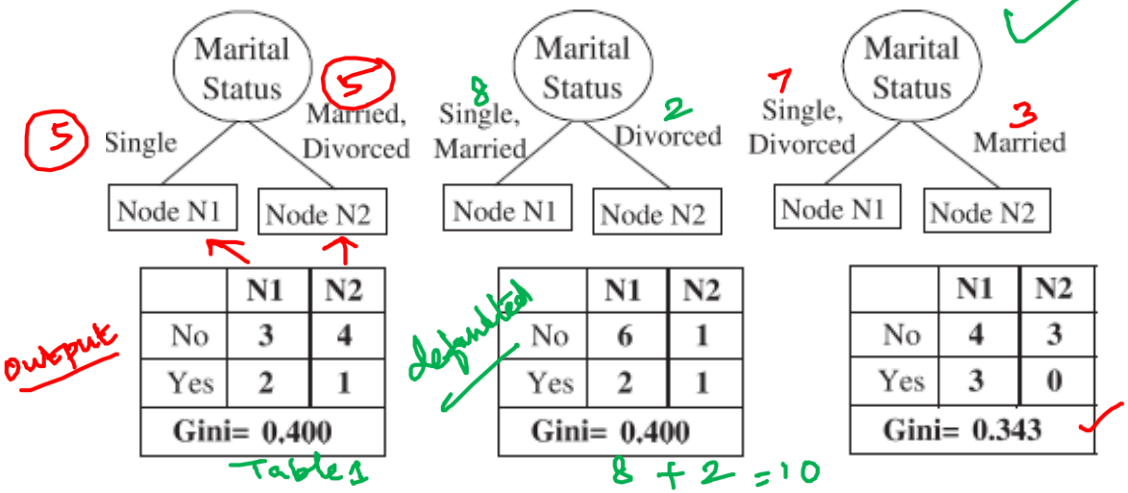
There are '3' cases

Splitting of Nominal Attributes : Gini Index : Binary Splits

ID	Home Owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes
6	No	Single	60000	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes
9	No	Married	75000	No
10	No	Single	90000	Yes

Splitting Attribute: MaritalStatus

3 possibilities



	N1	N2
No	3	4
Yes	2	1
Gini= 0.400		

	N1	N2
No	6	1
Yes	2	1
Gini= 0.400		

	N1	N2
No	4	3
Yes	3	0
Gini= 0.343		

Based on these results, Home Owner and the last binary split using Marital Status are clearly the best candidates, since they both produce the lowest weighted average Gini index

Compute Gini Index for Marital status (Table 1):

$$\text{Gini}_{N_1} = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$\text{Gini}_{N_2} = 1 - \left(\frac{4}{5}\right)^2 - \left(\frac{1}{5}\right)^2 = 0.32$$

$$\text{Weighted gini index}_{G_1} = \frac{5}{10}(0.48) + \frac{5}{10}(0.32) = 0.400$$

Gini Index for Marital status (Table 2):

$$\text{Gini}_{N_1} = 1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2 = 0.375$$

$$\text{Gini}_{N_2} = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$\text{Weighted gini index}_{G_2} = \frac{8}{10}(0.375) + \frac{2}{10}(0.5) = 0.400$$

Gini index for Marital Status (Table 3):

$$\text{Gini}_{N_1} = 1 - \left(\tfrac{4}{7}\right)^2 - \left(\tfrac{3}{7}\right)^2 = 0.4898$$

$$\text{Gini } N_2 = 1 - \left(\tfrac{3}{3}\right)^2 - \left(\tfrac{0}{3}\right)^2 = 0$$

$$\text{Weighted gini index}_{G_3} = \tfrac{7}{10}(0.4898) + \tfrac{3}{10}(0) = 0.343$$

Option 3 is the best node

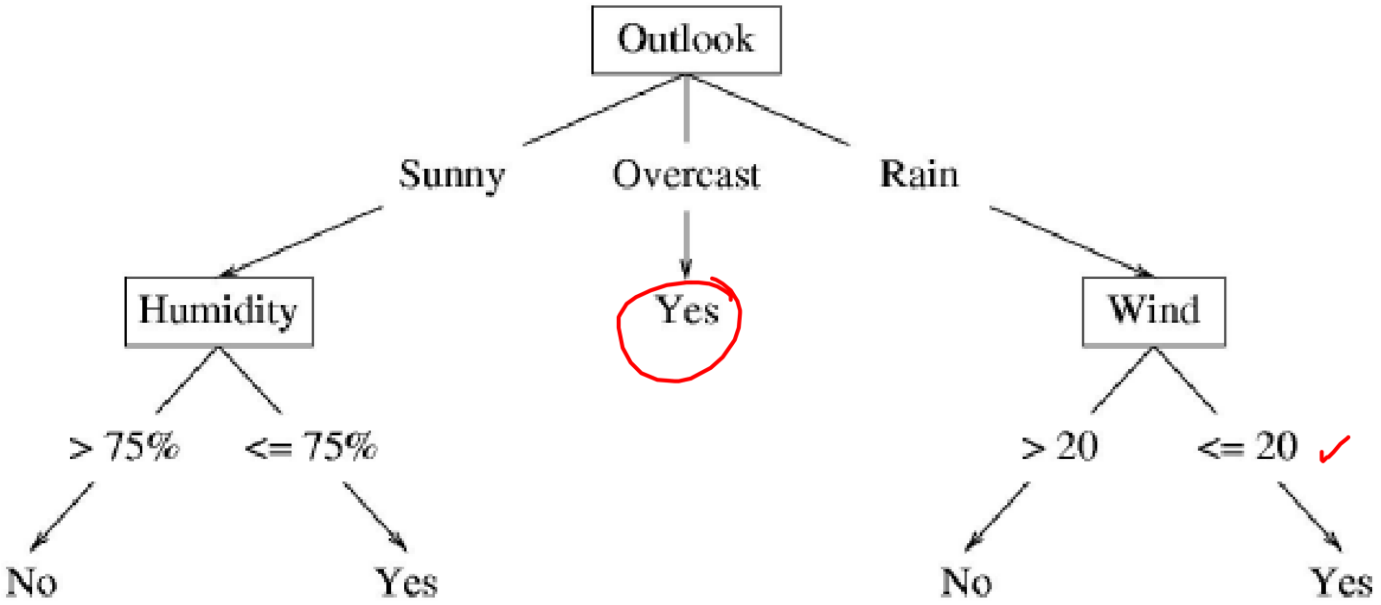
$$\text{Gain}_1 = \text{Gini (parent)} - \text{Gini}(G_1) = 0.420 - 0.4 = 0.02$$

$$\text{Gain}_2 = \text{Gini(parent)} - \text{Gini}(G_2) = 0.420 - 0.4 = 0.02$$

$$\text{Gain}_3 = \text{Gini(parent)} - \text{Gini}(G_3) = 0.420 - 0.343 = 0.07 \checkmark$$

Highest Gain is from 3rd table

Decision Trees – Dealing with Continuous Values



Decision Trees – Dealing with Continuous Values – Problem Type 5

- Given a continuous-valued attribute A, dynamically create a new attribute A_c

$$A_c = \begin{cases} \textbf{True } \textit{if } A < c, \\ \textbf{False } \textit{otherwise} \end{cases}$$

- How to determine threshold value c ?

Way 1: Multi-way Split

- Example. *Temperature* in the *PlayTennis* example
  - Sort the examples according to *Temperature*

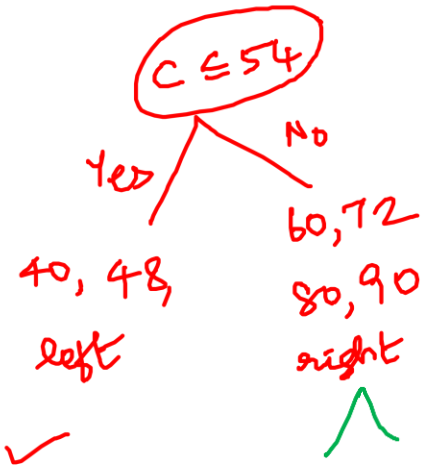
split needed

Temperature	40	48	|	60	72	80	|	90
PlayTennis	No	No	*54*	Yes	Yes	Yes	*85*	No

  - Determine candidate thresholds by averaging consecutive values where there is a change in classification: (48+60)/2=54 and (80+90)/2=85

📌 Step 2: Evaluate Splits

We'll evaluate **Gini Index** (or Info Gain) for each threshold and pick the best.



- **Split on $c = 54$:**
  - **Left (≤54):** 40, 48 → [No, No]
  - **Right (>54):** 60, 72, 80, 90 → [Yes, Yes, Yes, No]
  - Gini(Left) = 0 (pure class: No)
  - Gini(Right): 3 Yes, 1 No →

$$G = 1 - (3/4)^2 - (1/4)^2 = 1 - 0.5625 - 0.0625 = 0.375$$

  - Weighted Gini:

$$\frac{2}{6} \cdot 0 + \frac{4}{6} \cdot 0.375 = 0.25$$

- **Split on $c = 85$:**
  - **Left (≤85):** 40, 48, 60, 72, 80 → [No, No, Yes, Yes, Yes]
  - **Right (>85):** 90 → [No]
  - Gini(Right) = 0 (pure class: No)
  - Gini(Left): 2 No, 3 Yes →

$$G = 1 - (3/5)^2 - (2/5)^2 = 1 - 0.36 - 0.16 = 0.48$$

Weighted Gini:

$$\frac{5}{6} \cdot 0.48 + \frac{1}{6} \cdot 0 = 0.4$$

Best Split: $c = 54$
Since Gini = 0.25 < 0.4)

Step 3: Construct the Decision Tree

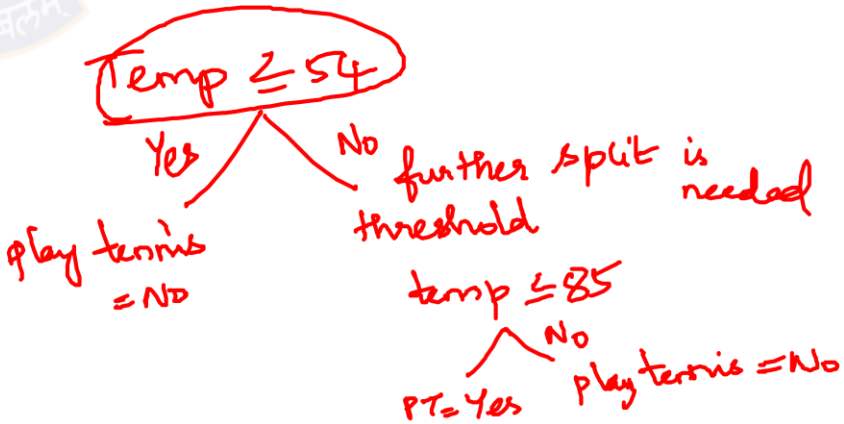
- Temperature ≤ 54?
- ├── Yes → PlayTennis = No
- └── No → (Split further)
- ├── Data: [60 (Yes), 72 (Yes), 80 (Yes), 90 (No)]
- └── Candidate threshold: (80+90)/2 = 85

Now split again with: $c = 85$

Temperature ≤ 85?
├── Yes → PlayTennis = Yes (60, 72, 80)
└── No → PlayTennis = No (90)

Final Decision Tree:

Temperature ≤ 54?
├── Yes → No
└── No →
Temperature ≤ 85?
├── Yes → Yes
└── No → No



Decision Trees – Dealing with Continuous Values – Problem Type 5

- Given a continuous-valued attribute *A*, dynamically create a new attribute A_c

$$A_c = \text{True } \textit{if } A < c, \text{ False } \textit{otherwise}$$

- How to determine threshold value *c* ?

60 70 75 85 90 95

O/P No No No Yes Yes Yes

find the threshold

Way 2: Binary Splits

- Example. *Annual Income*

Sort the examples according to *Annual Income*

Linearly scan all possible threshold, to determine best split point where the impurity is the least

ID	Home Owner	Marital Status	Annual Income	Defaulted?
1	Yes	Single	125000	No
2	No	Married	100000	No
3	No	Single	70000	No
4	Yes	Married	120000	No
5	No	Divorced	95000	Yes ✓
6	No	Single	60000 ✗	No
7	Yes	Divorced	220000	No
8	No	Single	85000	Yes ✓
9	No	Married	75000	No
10	No	Single	90000	Yes ✓

Decision Trees – Dealing with Continuous Values – Problem Type 5

Way 2: Binary Splits

- Example. *Annual Income*

Candidate split positions are identified by taking the midpoints between two adjacent sorted values: 55, 65, 72, and so on.
Extra split **before the smallest value** (and sometimes after the largest value).

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (in '000s)										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72.5	80	87.5	92.5	97.5	110	122.5	172.5	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0

Decision Trees – Dealing with Continuous Values – Problem Type 5

Way 2: Binary Splits

Split at 55

- **Left node (≤ 55)**: no data points → (Yes = 0, No = 0).
  - By definition, Gini(Empty) = 0.
- **Right node (> 55)**: all 10 samples → (Yes = 3, No = 7).
  - $p(Yes) = 3/10 = 0.3$, $p(No) = 7/10 = 0.7$.
  - $Gini(R) = 1 - (0.3^2 + 0.7^2)$

$$= 1 - (0.09 + 0.49) = 1 - 0.58 = 0.42$$

Weighted Gini for the split

$$Gini_{split} = \frac{0}{10} \cdot 0 + \frac{10}{10} \cdot 0.42 = 0.42$$

Split 97.5 → partitions:

- Left: ≤ 97.5 → incomes {60,70,75,85,90,95} → 6 samples: Yes = 3, No = 3.
- Right: > 97.5 → incomes {100,120,125,220} → 4 samples: Yes = 0, No = 4.

Compute Gini for each child:

Left node L:

$p_{Yes|L} = 3/6 = 0.5,\ p_{No|L} = 3/6 = 0.5$

$$Gini(L) = 1 - (0.5^2 + 0.5^2) = 1 - (0.25 + 0.25) = 0.5$$

Right node R:

$p_{Yes|R} = 0/4 = 0,\ p_{No|R} = 1$

$$Gini(R) = 1 - (0^2 + 1^2) = 0$$

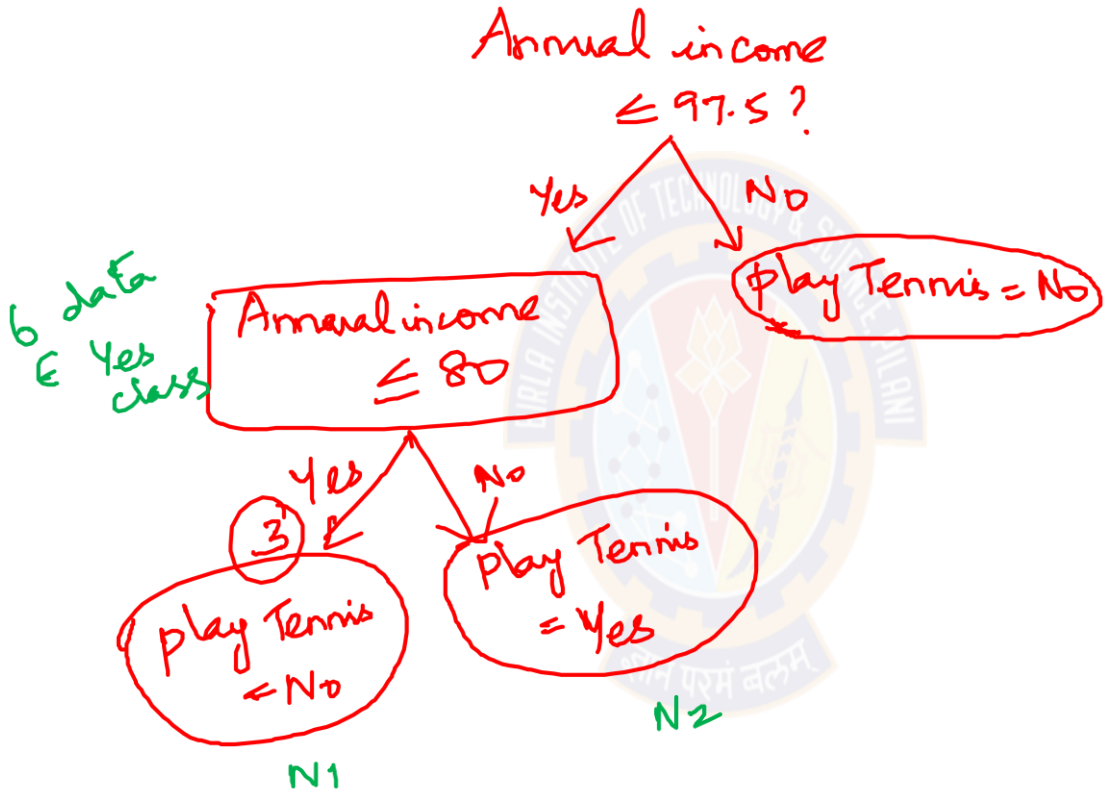
Weighted Gini of the split:

$$Gini_{split} = \frac{N_L}{N} Gini(L) + \frac{N_R}{N} Gini(R) = \frac{6}{10} \cdot 0.5 + \frac{4}{10} \cdot 0 = 0.6 \cdot 0.5 = 0.30$$

Sorted Values →

Split Positions →

Class	No	No	No	Yes	Yes	Yes	No	No	No	No
Annual Income (in '000s)										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72.5	80	87.5	92.5	97.5	110	122.5	172.5	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420


Annual income
≤ 97.5 ?
Yes
No
Play Tennis = No
6 data
6 Yes class
Annual income
≤ 80
Yes
No
3
play Tennis
= No
N1
Play Tennis
= Yes
N2

Classification Error

Classification Error (impurity measure)

$$\text{Error}(t) = 1 - \max_i p_i(t)$$

- Intuition:
  - $\max_i p_i(t)$ is the fraction of samples correctly classified if you always predict the majority class.
  - The rest, $1 - \max_i p_i(t)$, are misclassified → classification error.

Node N_1	Count
Class=0	0
Class=1	6

$\text{Error} = 1 - \max[0/6, 6/6] = 0$

Node N_2	Count
Class=0	1
Class=1	5

$\text{Error} = 1 - \max[1/6, 5/6] = 0.167$

Node N_3	Count
Class=0	3
Class=1	3

$\text{Error} = 1 - \max[3/6, 3/6] = 0.5$

Note: In practice, **Gini Index** or **Entropy** are often preferred over simple classification error when growing a decision tree, because classification error is **less sensitive to changes in class proportions**.

An alternative measure: gain ratio

- SplitInformation measures the entropy of S with respect to the values of A.
- The more uniformly dispersed the data the higher it is.

Let S be the set of examples and attribute A partition S into subsets $S_1, S_2, \ldots, S_c$ (one subset per value of A). Let $|S| = n$ and $|S_i|$ be the size of subset S_i.

$$\text{SplitInformation}(S, A) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \left(\frac{|S_i|}{|S|} \right)$$

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}$$

(where $\text{Gain}(S, A)$ is the standard information gain: $\text{Entropy}(S) - \sum_i \frac{|S_i|}{|S|} \text{Entropy}(S_i)$).

favour attribute with large number of unique values (Eg: student id)

- **Information Gain:** "How much does this attribute help me predict the target?"

- **Split Information:** "How fragmented and complex is the split created by this attribute?"

- **Gain Ratio:** Balances the two to ensure the chosen attribute is both predictive and simple enough to generalize.

Limitation of Information Gain: Ways to tackle

GainRatio penalizes attributes that split examples in many small classes such as CustomerID, *Date.*

Let $|S|=n$, *Customer ID* splits examples in n classes.

For the **Customer ID** attribute:

1. There are n branches (one for each ID).
2. In each branch, the size $|S_i|$ is **1**.
3. The total size $|S|$ is n.
4. Substituting these in, you get the term: $\frac{1}{n}\log_2\frac{1}{n}$.

$$SplitInformation(S, customerID) = -[(1/n \log_2 1/n)+\ldots+(1/n \log_2 1/n)] = -\log_2 1/n = \log_2 n$$

Compare with *A*, which splits data in two even classes:

$$SplitInformation(S, A) = -[(1/2 \log_2 1/2)+(1/2 \log_2 1/2)]$$
$$= -[-1/2 - 1/2] = 1$$

$$= -\left[\frac{10}{20}\log_2\frac{10}{20}+\frac{10}{20}\log_2\frac{10}{20}\right]$$

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Select the best attribute test condition among the following three attributes: **Gender, Car Type, and Customer ID**

Limitation of Information Gain: Ways to tackle: Problem Type 6

$$\text{Entropy(parent)} = -\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20} = 1.$$

If Gender is used as attribute test condition:

$$\text{Entropy(children)} = \frac{10}{20}\left[-\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10}\right] \times 2 = 0.971$$

$$\text{Gain Ratio} = \frac{1 - 0.971}{-\frac{10}{20}\log_2\frac{10}{20} - \frac{10}{20}\log_2\frac{10}{20}} = \frac{0.029}{1} = 0.029$$

If Car Type is used as attribute test condition:

$$\text{Entropy(children)} = \frac{4}{20}\left[-\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4}\right] + \frac{8}{20} \times 0 + \frac{8}{20}\left[-\frac{1}{8}\log_2\frac{1}{8} - \frac{7}{8}\log_2\frac{7}{8}\right] = 0.380$$

$$\text{Gain Ratio} = \frac{1 - 0.380}{-\frac{4}{20}\log_2\frac{4}{20} - \frac{8}{20}\log_2\frac{8}{20} - \frac{8}{20}\log_2\frac{8}{20}} = \frac{0.620}{1.52} = 0.41$$

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Select the best attribute test condition among the following three attributes: **Gender, Car Type, and Customer ID**

Car type → attribute

$$Gain(S, C_1) = -\frac{4}{20}\left[\left(\frac{3}{10}\log_2\frac{3}{10}\right)\right] \quad \text{(Family)}$$

$$-\frac{8}{20}\left[\left(\frac{7}{10}\log_2\frac{7}{10}\right)\right] \quad \text{(Luxury)}$$

$$= -\frac{4}{20}(-0.522) - \frac{8}{20}(-0.108)$$

$$= 0.1044 + (0.0432)$$

$$= 0.1476$$

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

$$\text{Gain ratio}(S, \text{Car type}) = \frac{}{1.52} = \frac{0.8524}{1.52} = 0.56$$

$$\text{Entropy}_{\text{Gender}} = \text{Weighted Entropy} = \frac{10}{20}(0.971) + \frac{10}{20}(0.971) = 0.971$$

Males (10) $\rightarrow$ 6 C_0, 4 C_1

Females (10) $\rightarrow$ 4 C_0, 6 C_1

$$\text{Entropy}_{\text{Males}} = -\frac{6}{10}\log_2\left(\frac{6}{10}\right) - \frac{4}{10}\log_2\left(\frac{4}{10}\right) = 0.971$$

$$\text{Entropy}_{\text{Females}} = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) = 0.971$$

$$\text{Gain}(S, \text{Gender}) = 1 - 0.971 = \text{Entropy}_{\text{parent}} - \text{Entropy}_{\text{children}}$$

$$= 0.029$$

$$\text{Split Information} = -\left[\frac{10}{20}\log_2\left(\frac{10}{20}\right) + \frac{10}{20}\log_2\left(\frac{10}{20}\right)\right] = 1$$

$$\therefore \text{Gain Ratio} = \frac{\text{Gain}(S, \text{Gender})}{\text{split info}} = 0.029$$

Limitation of Information Gain: Ways to tackle: Problem Type 6

Customer ID	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

Finally, if `Customer ID` is used as attribute test condition:

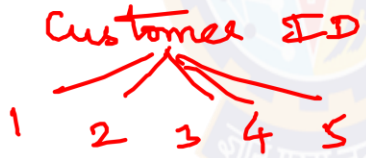
Weighted

$$\text{Entropy(children)} = \frac{1}{20}\left[-\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1}\right] \times 20 = 0$$

(C_0, C_1)

$$\text{Gain Ratio} = \frac{1-0}{-\frac{1}{20}\log_2\frac{1}{20} \times 20} = \frac{1}{4.32} = 0.23$$

Information Gain$_{ID}$ = 1

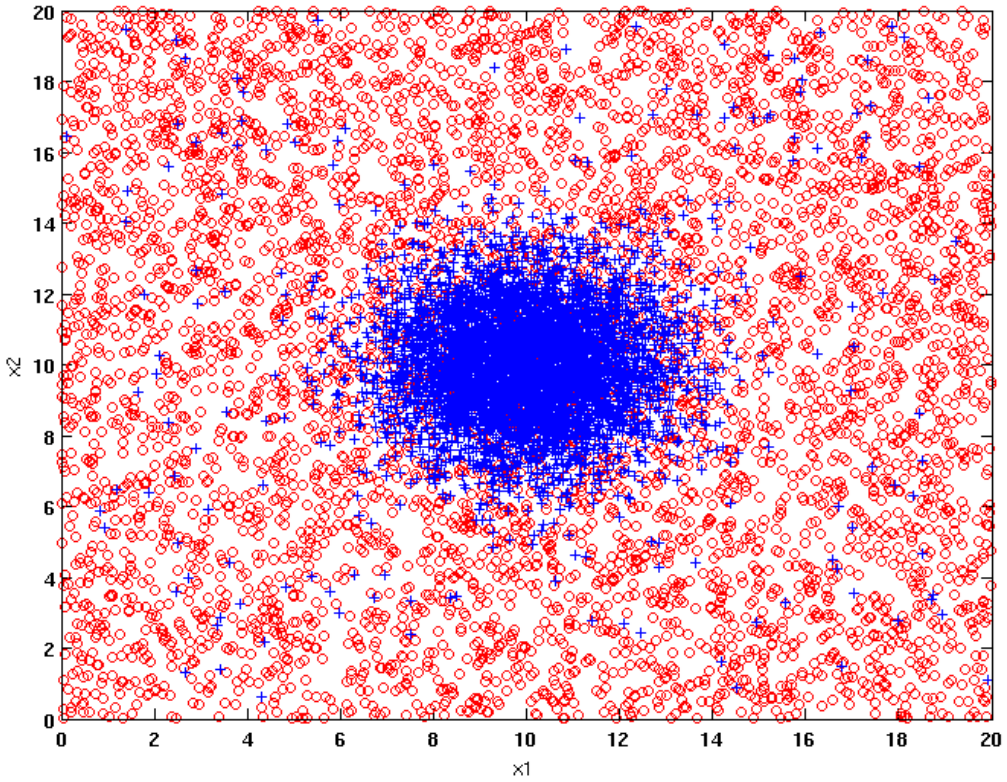


Select the best attribute test condition among the following three attributes: **Gender, Car Type, and Customer ID**

Inference : Thus, even though Customer ID has the highest information gain, its gain ratio is lower than Car Type since it produces a larger number of splits

Gain Ratio → splits that divide the data into small groups, giving more balanced & fair attribute selection used in C4.5 (successor of ID3) algorithm.

Example Data Set



Two class problem(class o and class +**):**

+ : 5200 instances

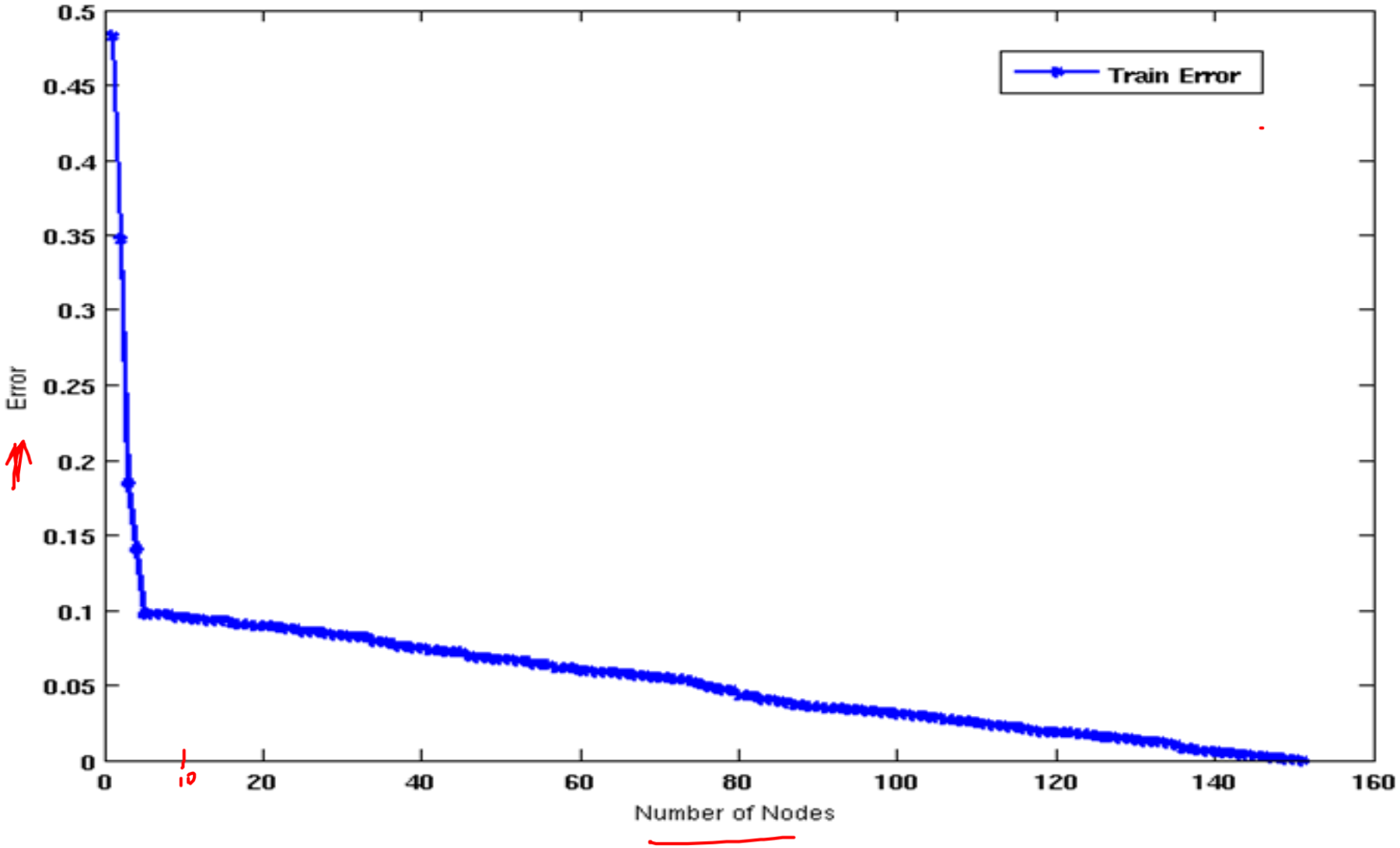
- **5000 instances generated from a Gaussian centered at (10,10)**
- **200 noisy instances added**

o : 5200 instances

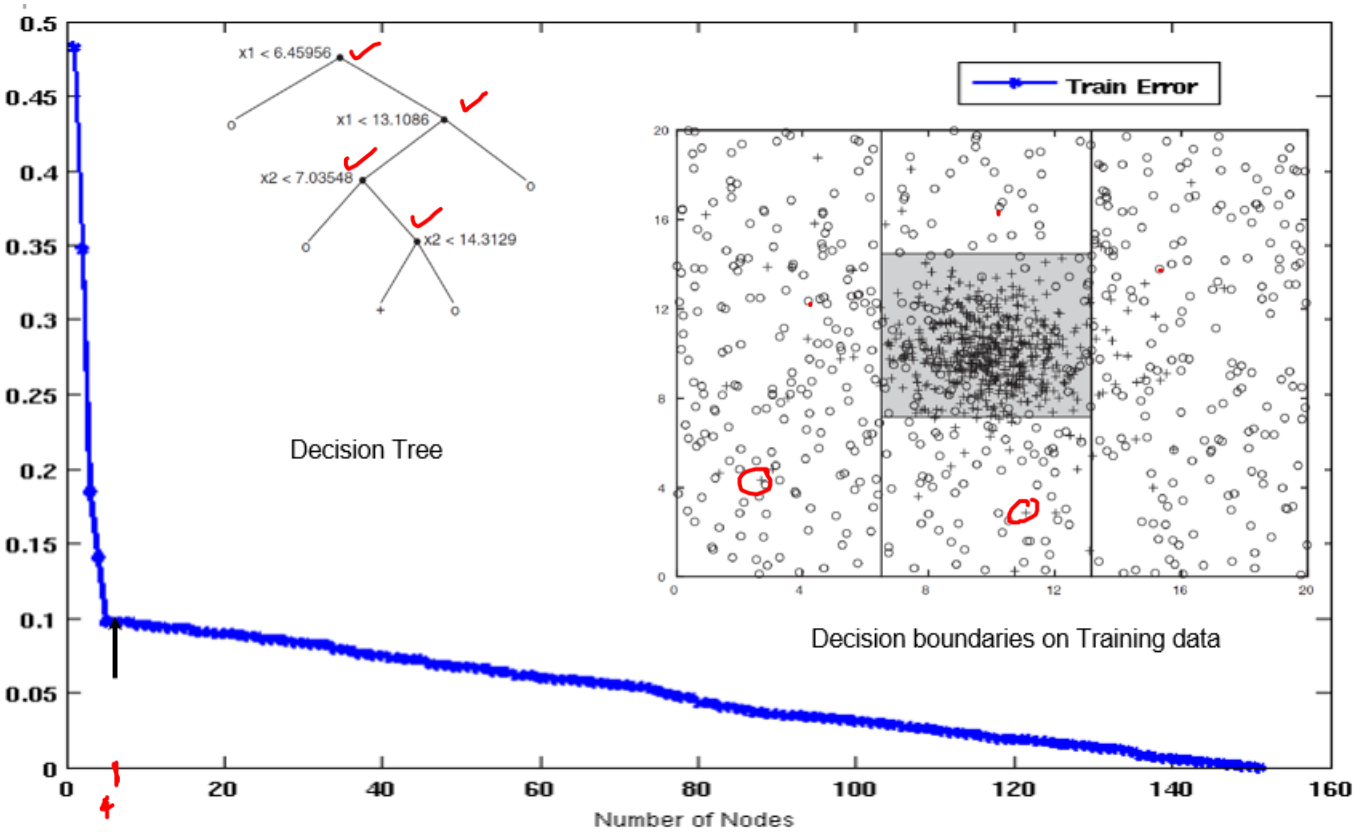
- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

Increasing number of nodes in Decision Trees



Decision Tree with 4 nodes



Low Complexity = Underfitting Risk

- The tree captures only **coarse patterns** in the data.
- It doesn't tightly follow the cluster of + symbols in the center.
- As a result, many + samples lie in areas predicted as o, and vice versa.
- This suggests a **bias problem**: the tree is **too simple** to capture the structure in the data.

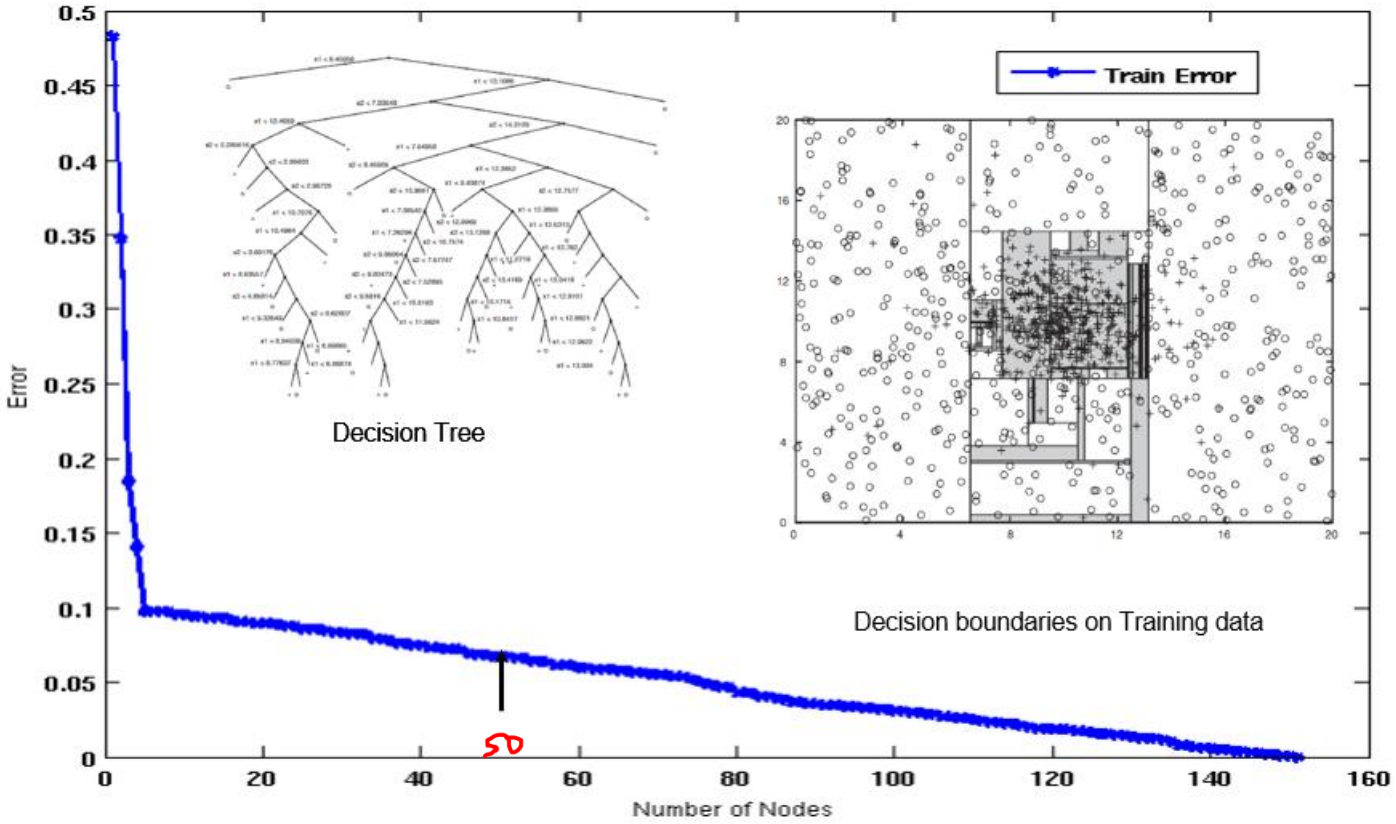
Connection to the Left-Side Plots

•The **decision tree diagram** (top-left inset) shows just a few splits:

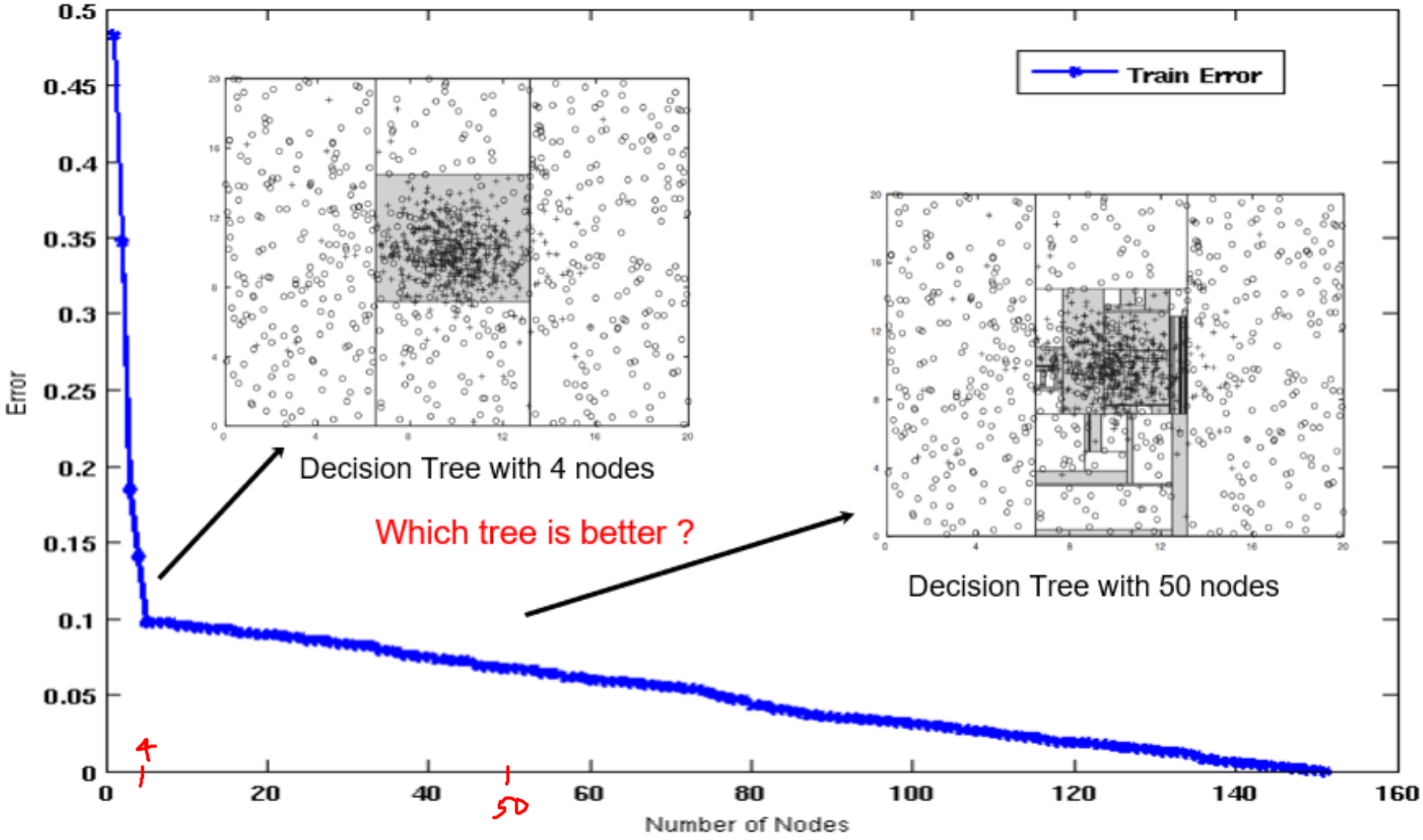
- `x1 < 6.46`
- `x1 < 13.11`
- `x2 < 7.04`
- `x2 < 14.31`

•These splits correspond directly to the visible boundaries in the right-hand graph.

Decision Tree with 50 nodes



Which tree is better? Problem Type 7 – Model Comparison



Which tree is better — the one with 4 nodes or 50 nodes?

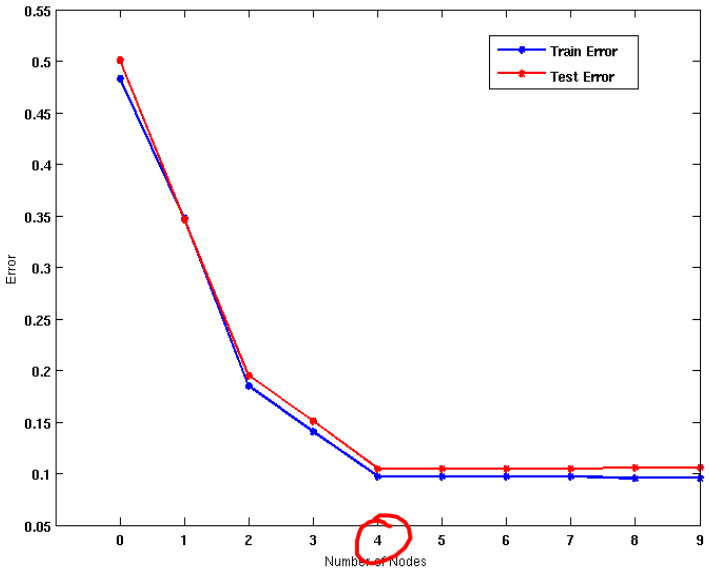
Answer:

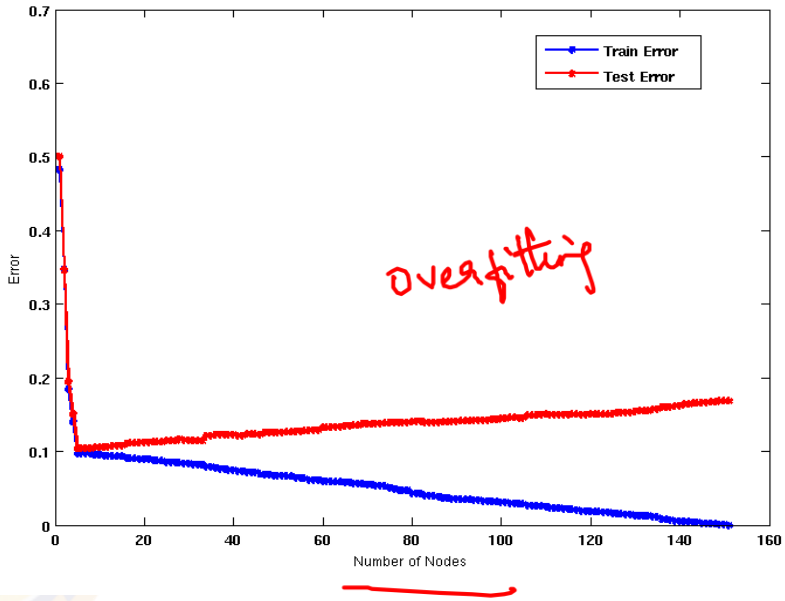
- **It depends on the trade-off between bias and variance, but most likely:**
- ✅ **The 50-node tree is better** — if your goal is to improve accuracy and the tree generalizes well (e.g., validated with test or cross-validation error).
- Captures the complex structure in the center.

🚫 The 4-node tree is too simple and **underfits** the data.

- When **interpretability** is more important than accuracy.
- In real-time systems where **speed** and **simplicity** are critical.

Model Overfitting



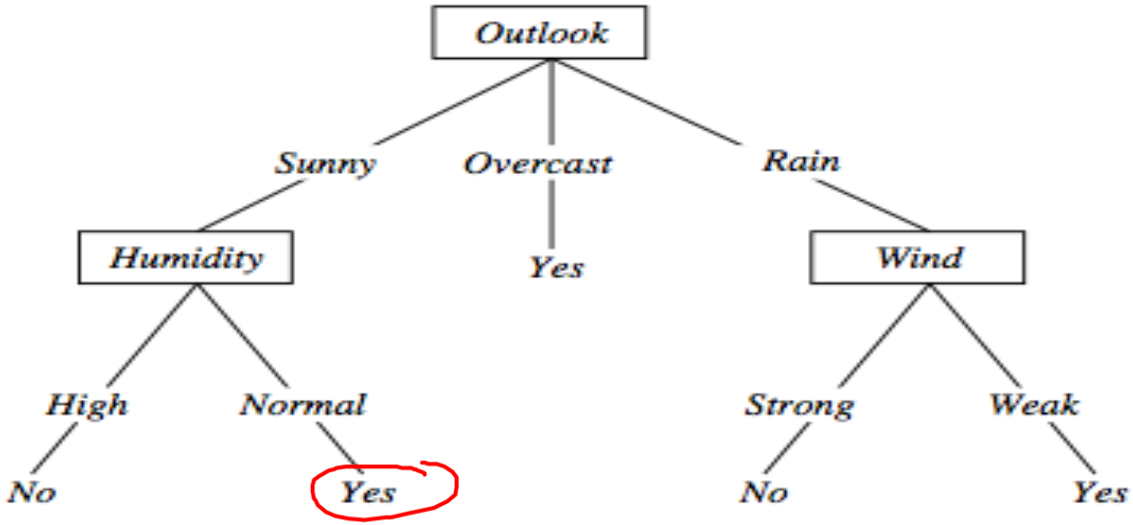


Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

no. of nodes is large

Overfitting in decision trees



⟨*Outlook=Sunny, Humidity=Normal, Wind=Strong, PlayTennis=No* ⟩

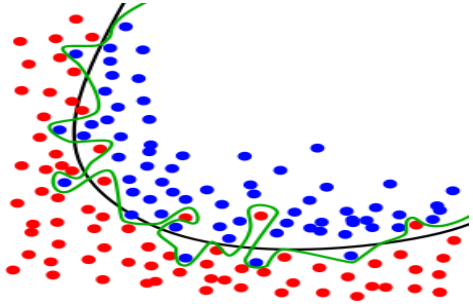
For "Sunny, Humidity = Normal", the current tree says **Yes**.
But this new training example says **No** → a **contradiction**.

New noisy example causes splitting of second leaf node.

- The tree looks “perfect” on training data.
- But on **new unseen data**, accuracy may drop because the tree has become too specific.

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, may reflect anomalies due to noise or outliers in the training data
  - Poor accuracy for unseen samples



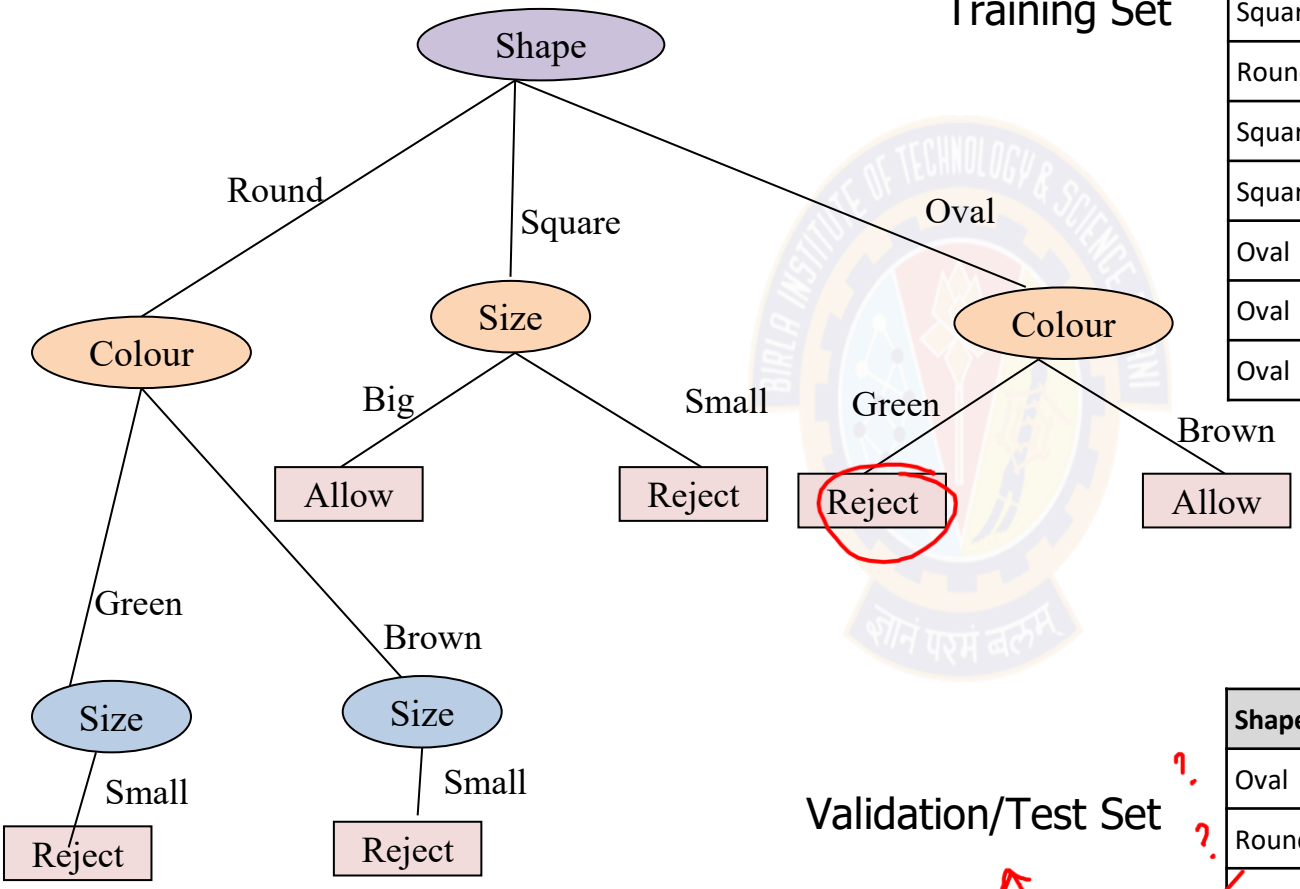
Approaches to tackle:

- Pruning
- Constraint on the depth of tree
- Constraint on the minimum nodes allowed at leaf

Over fitting

Assumed over fit model

- ID3



Training Set

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Validation/Test Set

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-pruning

- **Tree** growth is stopped early, before it fully classifies the training data perfectly.
- Instead of letting the tree grow until every training example is separated, the algorithm uses stopping conditions to decide when to stop splitting.

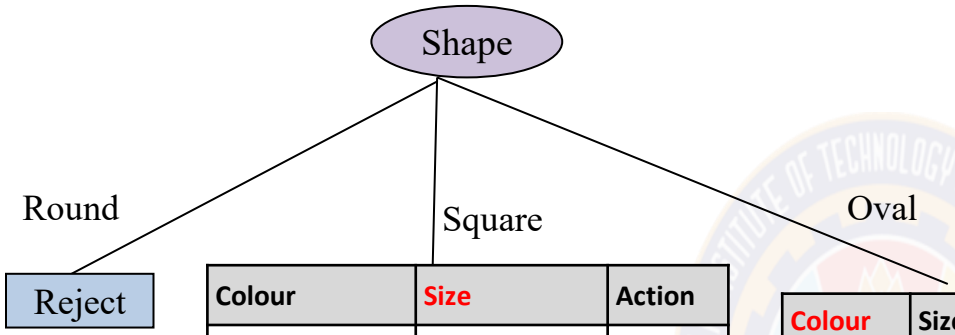
- Typical Stopping Conditions
  - Stop if the information gain (or Gini gain) is below a threshold.
  - Stop if the node has fewer than a minimum number of samples.
  - Stop if tree depth exceeds a maximum level.
  - Stop if improvement in accuracy/generalization is not significant.

Choosing the **right threshold** is tricky:
- Too strict → tree becomes too shallow → **underfitting**.
- Too loose → tree still grows too much → **overfitting**.

Pre-Pruning – Problem Type 8

- *Threshold:* ***Gain <=0.85***

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject



Colour	Size	Action
Black	Big	Allow
Brown	Big	Allow
Green	Big	Allow
Brown	Small	Reject

Gain = Size → **0.811**
Colour → 0.311

Colour	Size	Action
Green	Big	Reject
Brown	Small	Allow
Green	Small	Reject

Gain = Colour → **0.918**
Size → 0.251

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

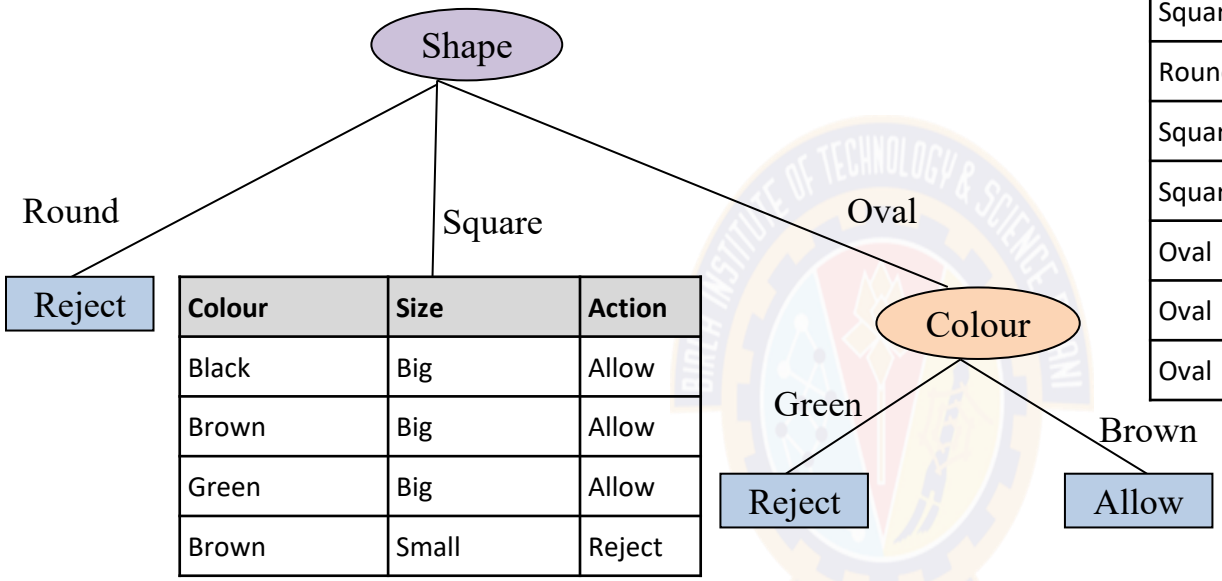
Square: 0.811 < 0.85 → **do not split** the square node (it becomes a leaf).
Oval: 0.918 ≥ 0.85 → **split on Colour**.

Idea : While construction, Prune the nodes whose gain is less than or equal to the threshold.

Pre-Pruning – Problem Type 8

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Threshold: Gain <=0.85



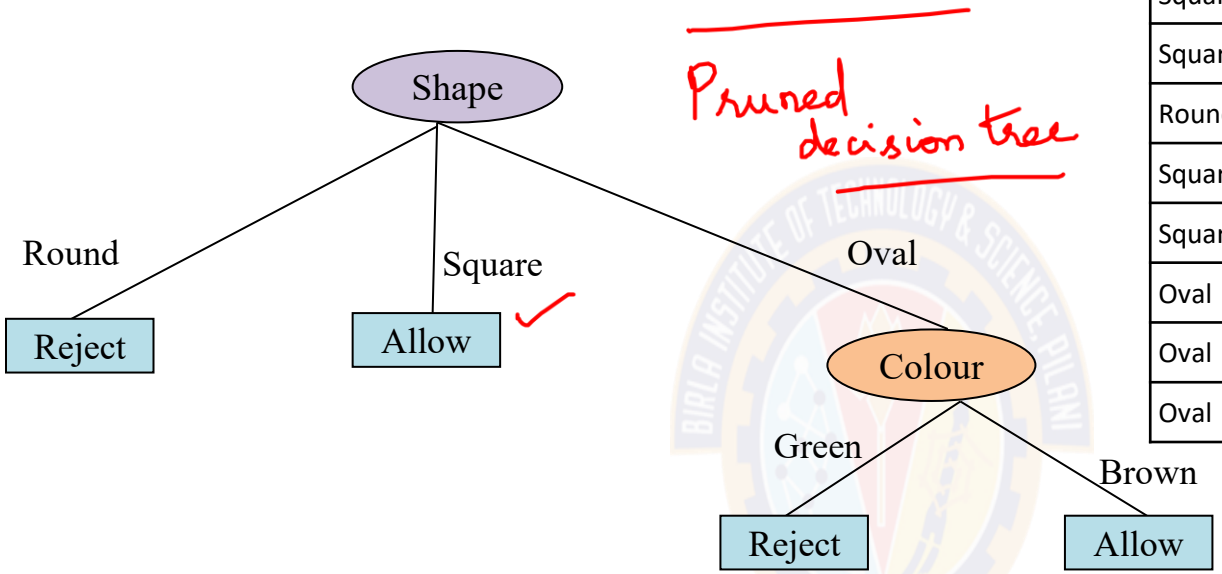
Colour	Size	Action
Black	Big	Allow
Brown	Big	Allow
Green	Big	Allow
Brown	Small	Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Pre-Pruning – Problem Type 8

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Threshold: Gain <=0.85



Pruned decision tree

Apply majority voting for converting the pruned subset of data into a class

3 cases allow

1 case Reject

Shape	Colour	Size	Action
Oval	Black	Small	Reject
Round	Brown	Big	Allow
Square	Brown	Big	Allow
Oval	Green	Small	Allow

Post-Pruning

Idea :

1. Grow the tree fully: Don't stop early — keep splitting until every node is pure (or minimal samples per leaf).
2. Scan the tree bottom-up
3. At every decision node
   - Retain the attribute node & evaluate it against the prune set / validation set(used to estimate the cost)
   - Remove the attribute node & reevaluate it with the same prune set
   - If there is a reduction in error, prune the node else retain the node
4. Repeat this in other branches of the tree

Error rate is the percentage of tuples misclassified

Purpose of This Analysis

The goal of this post-pruning analysis is to **simplify** the decision tree **without sacrificing predictive performance**.

By pruning unnecessary or weak subtrees (like Size or Colour in this case), we can:

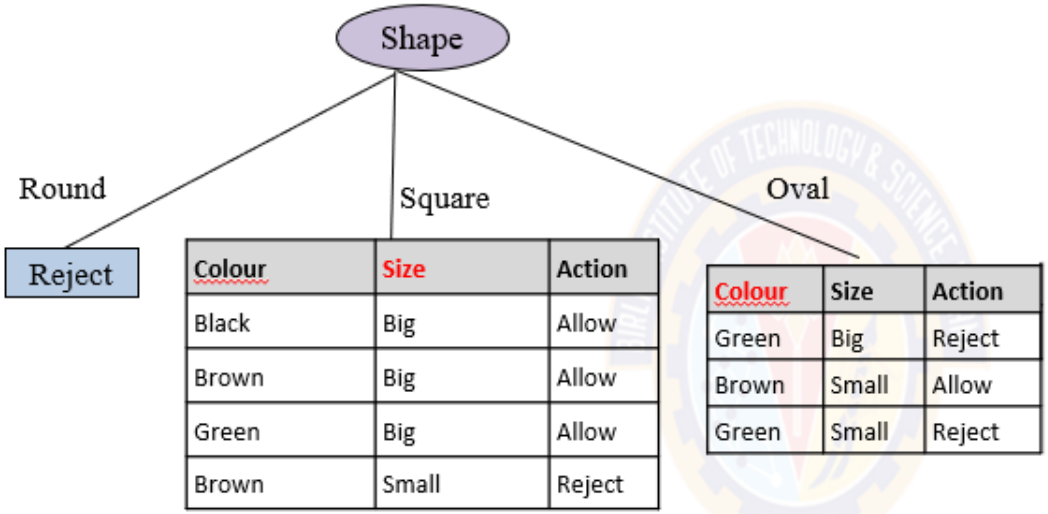
- Make the tree smaller
- Improve generalization
- Avoid overfitting

But we need to ensure pruning doesn't cause too many wrong predictions.

Post-pruning means:
"Remove some internal nodes if replacing them with a leaf improves or does not reduce accuracy."
Instead of keeping deeper splits, we **replace a subtree with a single decision (majority class)**.

Post-Pruning – Problem Type 9

Assumed model from the output of one intermediate iteration of the decision tree building



Colour	Size	Action
Black	Big	Allow
Brown	Big	Allow
Green	Big	Allow
Brown	Small	Reject

Colour	Size	Action
Green	Big	Reject
Brown	Small	Allow
Green	Small	Reject

Should we **remove the Size node** and directly predict for "Square"?

Training Dataset

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Prune Set

unseen data

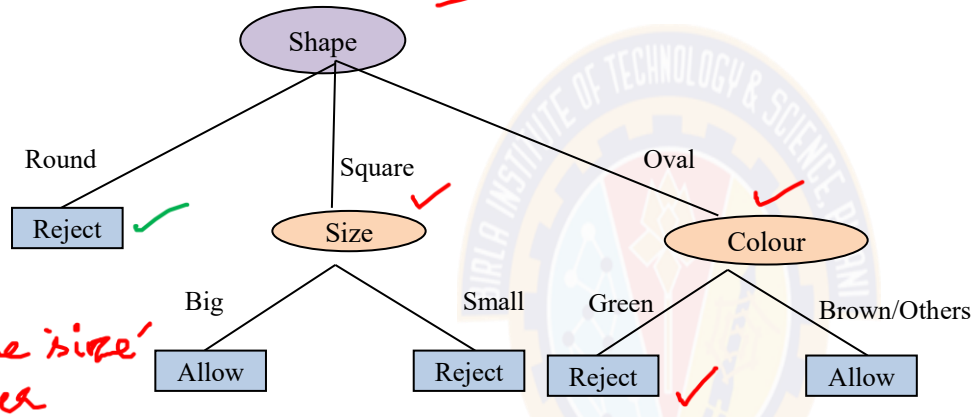
Shape	Colour	Size	Action
Oval	Black	Small	Allow
Round	Brown	Big	Reject
Square	Brown	Big	Allow
Oval	Green	Small	Reject

Not answered

Post-Pruning – Problem Type 9

Shape = square → ?? (leaf node)

Allow (majority)



removing the 'size' node under square branch

Shape	Colour	Size	Action
Round	Green	Small	Reject
Square	Black	Big	Allow
Square	Brown	Big	Allow
Round	Brown	Small	Reject
Square	Green	Big	Allow
Square	Brown	Small	Reject
Oval	Green	Big	Reject
Oval	Brown	Small	Allow
Oval	Green	Small	Reject

Prune Size &Predict	PruneColour & Predict	AboveTree's Prediction	Shape	Colour	Size	Action
Allow	Reject	Allow	Oval	Black	Small	Allow
Reject	Reject	Reject	Round	Brown	Big	Reject
Allow	Allow	Allow	Square	Brown	Big	Allow
Reject	Reject	Reject	Oval	Green	Small	Reject

Applying It in the Table (Example)

- Let's verify the first row:

Shape	Colour	Size	Action	Prune Size & Predict	Prune Colour & Predict
Oval	Black	Small	Allow	Allow	**Reject** ❌

- Shape = Oval → If we prune **Colour**, we predict **Reject**.
- True action is **Allow** → so pruning **Colour** gives a **wrong** prediction.

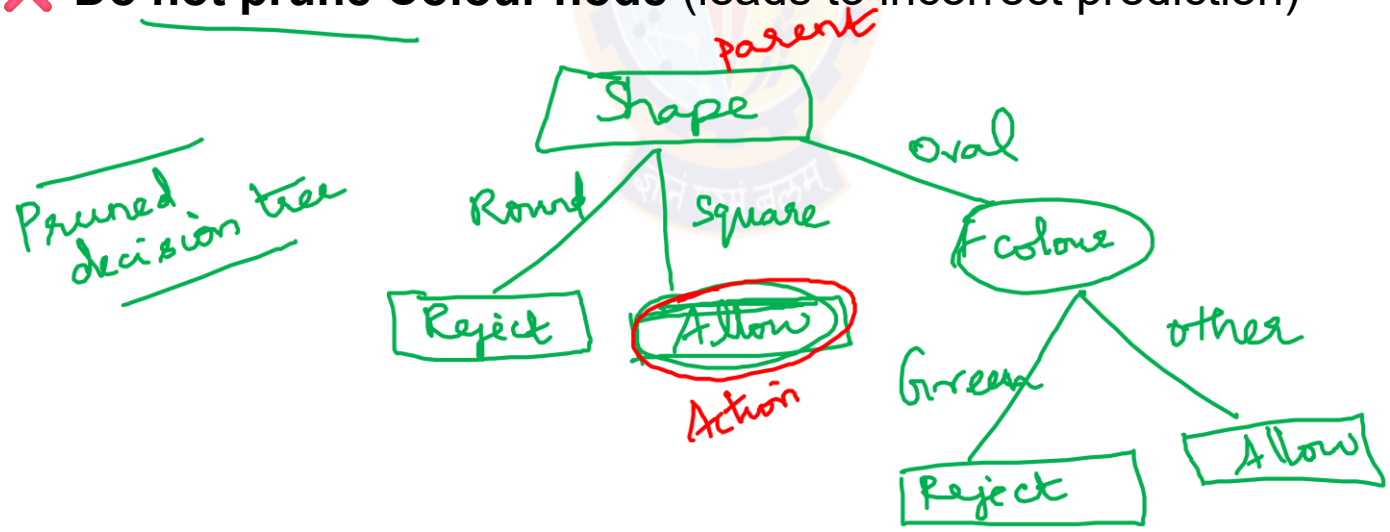
Decision: To Prune or Not?

From this analysis:

- **Pruning the Colour node under Oval** causes an error (Row 1).
- **Pruning the Size node under Square** does **not** cause any error in these test samples.

So:

- ✅ **Prune Size node** (safe to prune)
  ❌ **Do not prune Colour node** (leads to incorrect prediction)



Minimum Description Length Principle

The best decision tree is the one that minimizes the total description length of:

- **The tree itself (its structure: number of nodes, attributes used, thresholds)**
- **The data when encoded using the tree (the errors or exceptions that the tree cannot perfectly explain).**

$$
\begin{aligned}
h_{MAP} &= \arg\max_{h \in H} P(D|h)P(h) \\
&= \arg\max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\
&= \arg\min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \quad (1)
\end{aligned}
$$

Interesting fact from information theory:

The optimal (shortest expected coding length) code for an event with probability p is $-\log_2 p$ bits.

So, interpret (1):

- $L_{C1}(h) = length(h) = -\log_2 P(h)$
- $L_{C2}(D|h) = length(misclassifications) = -\log_2 P(D|h)$

→ prefer the hypothesis that minimizes

$length(h) + length(misclassifications)$

Bayes' Rule

Bayes' theorem gives us a way to update our belief about a hypothesis $h$after observing data D:

$$P(h \mid D) = \frac{P(D \mid h) \cdot P(h)}{P(D)}$$

Already happened → Total prob. formula

Where:

- $P(h \mid D)$is the **posterior probability** of the hypothesis $h$given data D.
- $P(D \mid h)$is the **likelihood** — how well the hypothesis explains the data.
- $P(h)$is the **prior** — how probable the hypothesis is *before* seeing data.
- $P(D)$is the **marginal likelihood** — overall probability of the data.

Minimum Description Length Principle

- MDL principle provides a way of trading off hypothesis complexity for the number of errors committed by the hypothesis.
- May select a shorter hypothesis that makes a few errors over a longer hypothesis that perfectly classifies the training data.
- Provides one method for dealing with the issue of overfitting the data
  - Large complex trees can memorize training data → low training error but poor generalization.

Minimum Description Length Principle

Occam's razor: prefer the shortest hypothesis (In decision trees, shorter trees are better, provided they still classify the data reasonably well.)

MDL: prefer the hypothesis *h* that minimizes

$$h_{MDL} = \underset{h \in H}{\operatorname{argmin}} L_{C_1}(h) + L_{C_2}(D|h)$$

$C_1 \rightarrow$ code 1

$C_2 \rightarrow$ code 2

where $L_C(x)$ is the description length of *x* under encoding *C*

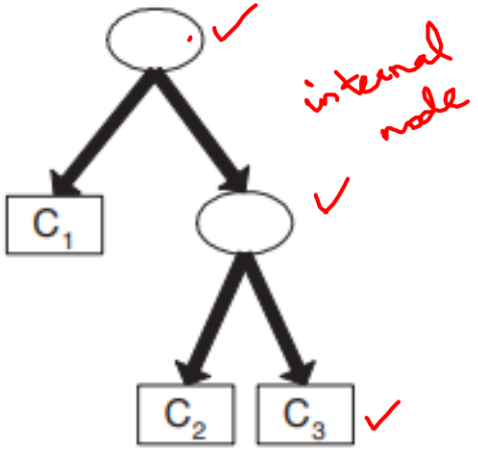
Example: H = decision trees hypothesis, D = training data labels

- $L_{C1}(h)$ is # bits to describe tree *h*
- $L_{C2}(D|h)$ is # bits to describe *D* given *h*
  - Note $L_{C2}(D|h) = 0$ if examples classified perfectly by *h*. Need only describe exceptions
- Hence h_{MDL} trades off tree size for training errors

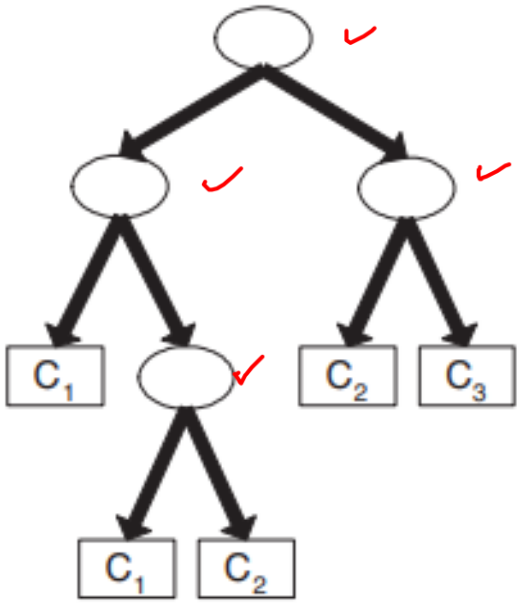
Minimum Description Length Principle
Problem Type 10

Consider the decision trees shown in Figure . Assume they are generated from a data set that contains 16 binary attributes and 3 classes, C_1, C_2, and C_3. Compute the total description length of each decision tree according to the minimum description length principle.

Which decision tree is better, according to the MDL principle?



(a) Decision tree with 7 errors



(b) Decision tree with 4 errors

Minimum Description Length Principle Problem Type 10

The **Minimum Description Length (MDL)** principle says:

> The best model is the one that minimizes the total description length of the model **plus** the data given the model.

For a decision tree:

$$L(T) = L(\text{Tree Structure}) + L(\text{Errors})$$

- **Tree structure cost** → how many bits are needed to describe the tree itself (internal nodes + leaf labels).
- **Error cost** → how many bits are needed to encode the training errors (the examples that the tree misclassifies).

The tree that gives the **shorter description length** is preferred, even if it makes more errors, because the model must balance **simplicity** vs. **accuracy**.

Minimum Description Length Principle Problem Type 10

Step 1. Cost of Internal Nodes

$\log_2(16) = \log_2(2^4)$

$= 4 \log_2 2$

Each internal node stores which **attribute** is used to split.

- There are 16 binary attributes → need $\log_2(16) = 4$ bits to encode one attribute.
- So each **internal node = 4 bits.** ✓

Step 2. Cost of Leaf Nodes

Each leaf stores a **class label**.

- There are 3 classes (C_1, C_2, C_3) → need $\lceil \log_2(3) \rceil = 2$ bits to encode the label.
- So each **leaf = 2 bits.** ✓

Step 3. Cost of Errors

When the decision tree misclassifies examples, we must encode them.

- There are n training examples.
- To identify which example was misclassified, we need $\log_2(n)$ bits per error.

So each **error** = $\log_2(n)$ **bits.** ✓

Minimum Description Length Principle
Problem Type 10

Step 4. Tree (a)

- Internal nodes = 2 → $2 \times 4 = 8$ bits
- Leaves = 3 → $3 \times 2 = 6$ bits
- Errors = 7 → $7 \times \log_2(n)$ bits

$$L(T_a) = 8 + 6 + 7\log_2(n) = 14 + 7\log_2(n)$$

Step 5. Tree (b)

- Internal nodes = 4 → $4 \times 4 = 16$ bits
- Leaves = 5 → $5 \times 2 = 10$ bits
- Errors = 4 → $4 \times \log_2(n)$ bits

$$L(T_b) = 16 + 10 + 4\log_2(n) = 26 + 4\log_2(n)$$

Step 6. Compare

Tree (a) better than Tree (b) if:

$$14 + 7\log_2(n) < 26 + 4\log_2(n)$$

$$3\log_2(n) < 12 \quad \Rightarrow \quad \log_2(n) < 4 \quad \Rightarrow \quad n < 16$$

$7\log_2(n) - 4\log_2(n) < 26 - 14$

$3\log_2(n) < 12$

Minimum Description Length Principle
Problem Type 10

Answer:

Because there are 16 attributes, the cost for each internal node in the decision tree is:

$$\log_2(m) = \log_2(16) = 4$$

Furthermore, because there are 3 classes, the cost for each leaf node is:

$$\lceil \log_2(k) \rceil = \lceil \log_2(3) \rceil = 2$$

The cost for each misclassification error is $\log_2(n)$.

The overall cost for the decision tree (a) is $2\times4+3\times2+7\times\log_2 n = 14+7\log_2 n$ and the overall cost for the decision tree (b) is $4\times4+5\times2+4\times5 = 26+4\log_2 n$. According to the MDL principle, tree (a) is better than (b) if $n < 16$ and is worse than (b) if $n > 16$.

Applications

Suited for following classification problems:

- Applications whose Instances are represented by attribute-value pairs.
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain missing attribute values

Real world applications

- Biomedical applications
- Manufacturing
- Banking sector
- Make-Buy decisions

Practice Exercises (for Students)

1. Which of the below discretization best suits to transform the attribute "Annual Income" for decision tree construction? Use entropy as decision criteria.
   - Annual Income (<=85k, >85k & <=200k, >200k)
   - Annual Income (<=90k, >90k)

2. If binary split is recommended for attribute "Marital Status" which of the combination of splits best fits? Justify your comment using entropy.

3. Use the results of part 1) & part 2) and Build a decision tree classifier using ID3 algorithm ie., Information gain and entropy measures. Grow the complete decision tree.
4. Use the test data to post prune the built tree. Try to prune atleast two internal nodes and choose the best of the trials

binary categorical continuous class

Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Test Data / Prune Set/ Validation Set			
Owner	Marital	Income	Default
No	Single	85k	No
Yes	Single	100k	Yes
Yes	Divorced	70k	No
No	Married	90k	Yes

References

References

T1 - Chapter # 3 (Tom M. Mitchell, Machine Learning)

R1 – Chapter # 3 (PANG-NING TAN, MICHAEL STEINBACH, VIPIN KUMAR, Introduction To Data Mining, Pearson, 2nd Edition.)

Additional References

Decision Tree

- https://www.youtube.com/watch?v=eKD5gxPPeY0&list=PLBv09BD7ez_4temBw7vLA19p3tdQH6FYO&index=1

Overfitting

- https://www.youtube.com/watch?time_continue=1&v=t56Nid85Thg
- https://www.youtube.com/watch?v=y6SpA2Wuyt8

Random Forest

- https://www.stat.berkeley.edu/~breiman/RandomForests/

Thank you !